**MEC ENG 102B: Mechatronics Design – Fall 2022: Final Project**

**Team 5: Youssef Elias, Nhi Phan, Miguel Salazar-Rivera, Martin Beshara**

***- An updated version of the "Opportunity" that this device addresses.***

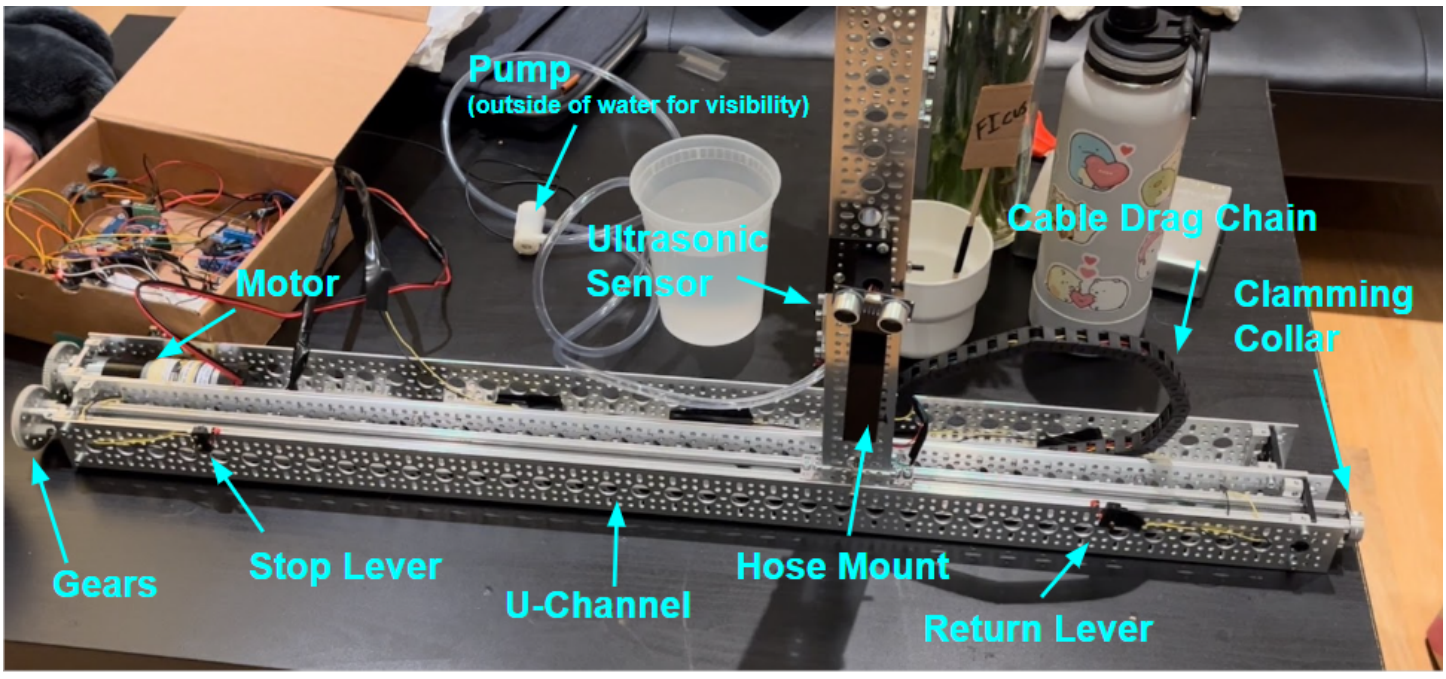
The purpose of our project is to address individuals who cannot attend to watering their plants. Plant care has been a significant challenge due to the need for physical presence and consistent daily watering. Therefore, to solve this issue, we developed the system *Planty Bot*, which will automatically water all plants in a household at specified intervals based on the user's preference. Our product is a solution for all plant owners enjoying vacations or immersing in the workforce, including students and disabled people, allowing them not to be further concerned about how and when to tend to their plants throughout their daily lives.

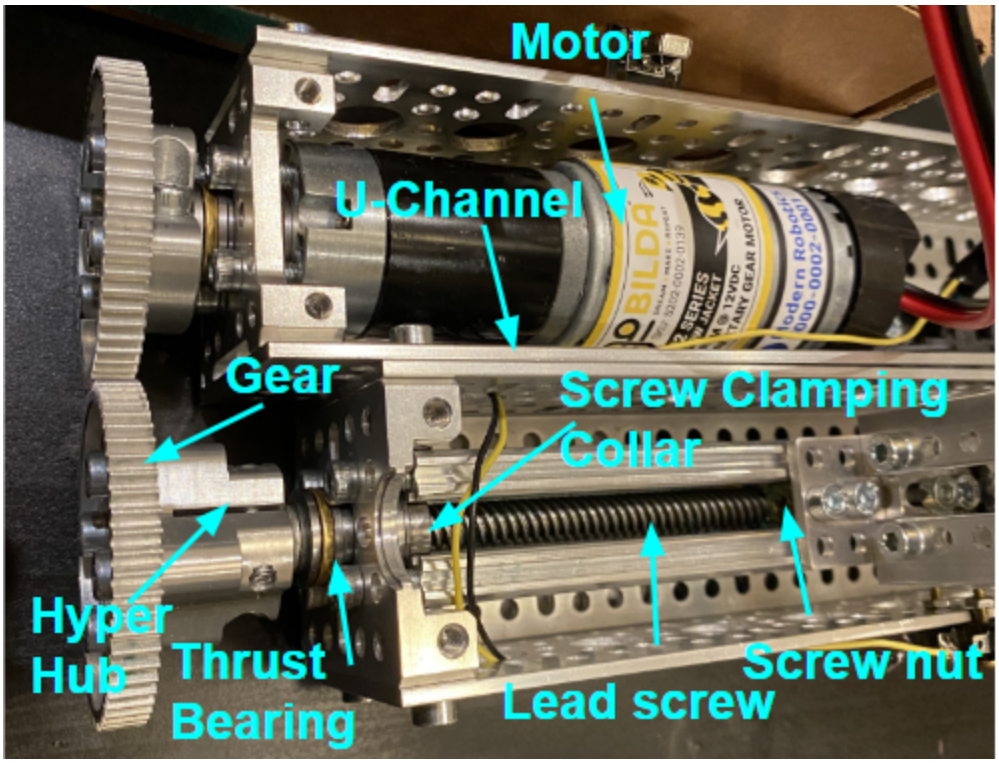
***- The device's high-level strategy & discuss/compare our initial desired functionality to our achieved specifications.***

At the start of our project, we had a high-aim plan for what to implement and how to achieve it. Initially, we decided to develop a device that freely moves in a designated area, detecting any plant through a camera and image-processing techniques. It would water the plant with a moving tank coupled with a specified flow rate, which could differ based on each pot. The watering was to be achieved through an arm freely moving vertically and horizontally toward the plant's direction, which would have been decided based on the pot's size. However, such a device would require three degrees of freedom, requiring an advanced, high-level electronic system that we found challenging to create. Therefore, we settled to develop a simpler model, which will still address the main objective but with less complexity relative to the original design.

Currently, our final product is a device that moves with one degree of freedom: back-and-forth movement through a linear track. Therefore, it is a constrained setup of a straight path for aligned and relatively equal-size pots, where plant detection is achieved through an ultrasonic sensor instead and watering is performed through a steady flow rate. Additionally, in place of incorporating a water tank within the system, which could cause loading and leakage, we placed the tank outside with an immersed pump connected to a hose and a 3D-printed mount. Finally, we embedded a static arm instead of a moving one to optimize the watering performance and minimize potential errors due to background noise.

***- A photo showing the integrated physical device, fully assembled and with labels showing where actuators and sensors are located.***






**- *Function-critical calculations*:**

*Gear Load*:

$$|F_y| = \frac{\tau}{r} = \frac{9\ N \cdot m}{48\ mm \cdot \frac{10^{-3} m}{1\ mm}} = 187.5\ N$$

$$tan(\theta) = \frac{F_x}{F_y} \rightarrow F_x = F_y \cdot tan(\theta) = (187.5\ N) \cdot tan(20) = 68.24\ N$$

$$F = \sqrt{F_x^2 + F_y^2} = \sqrt{(187.5\ N)^2 + (68.24\ N)^2} = 199.532\ N$$

*Torque required to move the body through the thread*:

$T_R = \frac{F \cdot d_m}{2} \cdot \frac{l + \pi\mu d_m}{\pi d - \mu l}$, where $F$ is the force applied to each gear teeth, μ is the coefficient of static friction between the gears, $d_m$ is the diameter of the gears, and $l$ is the length of the thread.

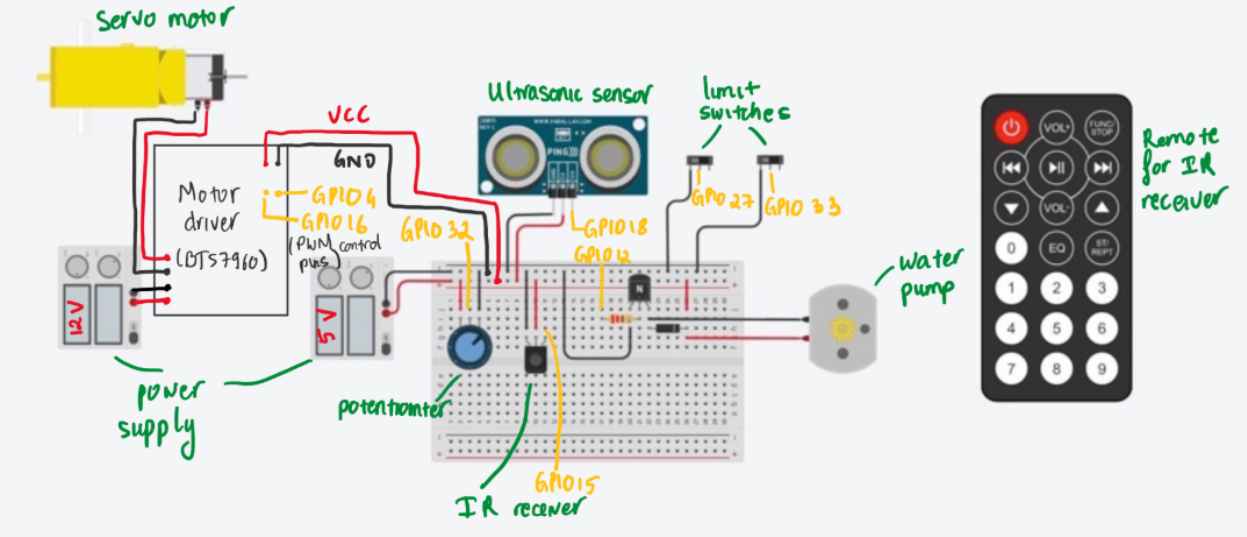
$$T_R = \frac{F \cdot (8\times10^{-3})}{2} \cdot \frac{38.1 \times 10^{-3} + \pi \times 0.8 \times (8\times10^{-3})}{\pi \times 8\times10^{-3} - 0.8 \cdot (38.1 \times 10^{-3})} = (0.03\ m) \cdot F = 0.0435\ m \cdot 199.532\ N$$

$T_R = 8.689\ N \cdot m$. This torque value will produce a force value of 300 N on the lead screw threads, which won't bend/break the threads.

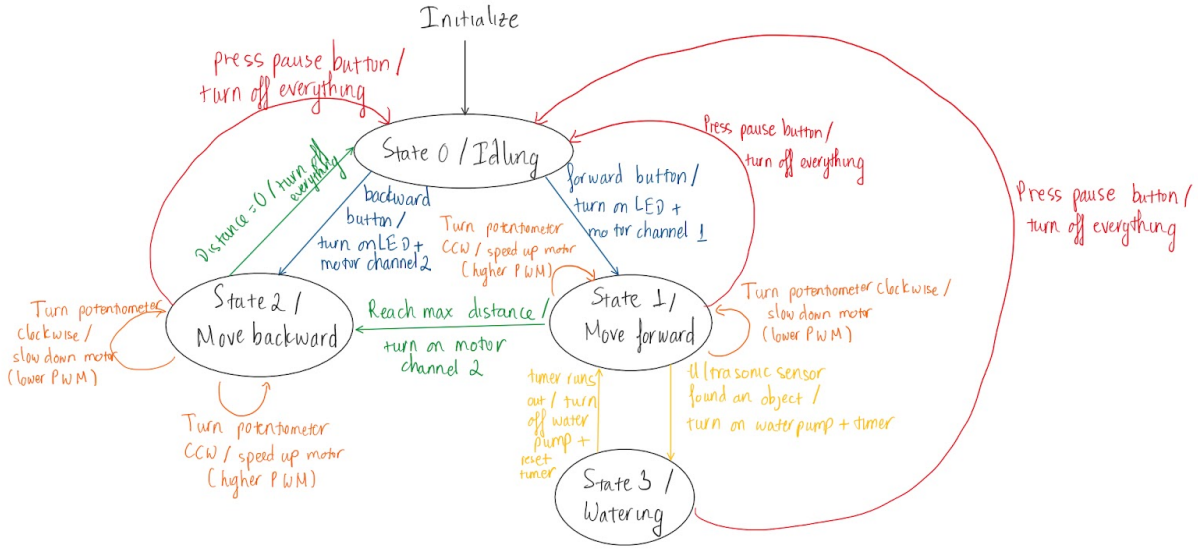
The lead screw thread gives 8 mm of linear movement per rotation. By experimentation, we get a result of a linear velocity of 8 mm per 1.5 seconds. Therefore, we get 1/1.5 = 2/3 revolutions per second.

$rpm = \frac{Linear\ Velocity}{Lead} = \frac{8/1.5\ mm/s}{8\ mm/rotation} = 40\ rpm$, where this value is the result of maximum voltage of 12 V.

**- *Circuit and State-Transition Diagrams*:**




*Circuit Diagram*




*State-Transition Diagram*

**- *Reflection and advice for future ME 102B students*.**

Obviously, starting early and mapping ideas with group members is critical. However, the real tricky part arises when members employ complex methodologies and ambitious goals, which happened with us. We recommend shifting the focus to developing relatively agile methods but with a sturdy, solid foundation, perfecting the basics of mechatronics. Finally, discussing your ideas with the staff is of utmost importance due to their wealth of experience in witnessing many group projects and interacting with the industry side.

***Bill of Materials***:

| Item Name | Quantity | Cost | Vendor & Link |
|---|---|---|---|
| 1/2" ID, .75" OD unshielded ball bearings | 1 | $6.27 | McMaster Carr: https://www.mcmaster.com/60355K505/ |
| 2807 Series Stainless Steel Shim (4mm ID x 7mm OD, 0.25mm Thickness) - 12 Pack | 1 | $1.99 | ServoCity: https://www.servocity.com/2807-series-stainless-steel-shim-4mm-id-x-7mm-od-0-25mm-thickness-12-pack/ |
| 1516 Series 8mm REX Standoff (M4 x 0.7mm Threads, 43mm Length) - 4 Pack | 2 | $4.04 | ServoCity: https://www.servocity.com/1516-series-8mm-rex-standoff-m4-x-0-7mm-threads-43mm-length-4-pack/ |
| 1201 Series Quad Block Pattern Mount (43-5) | 2 | $6.99 | ServoCity: https://www.servocity.com/1201-series-quad-block-pattern-mount-43-5/ |
| 1205 Series Dual Block Mount (1-5) - 2 Pack | 2 | $5.99 | ServoCity: https://www.servocity.com/1205-series-dual-block-mount-1-5-2-pack/ |
| 1521 Series 6mm ID Spacer (8mm OD, 4mm Length) - 4 Pack | 2 | $2.49 | ServoCity: https://www.servocity.com/1521-series-6mm-id-spacer-8mm-od-4mm-length-4-pack/ |
| 1201 Series Quad Block Pattern Mount (27-1) | 2 | $6.99 | ServoCity: https://www.servocity.com/1201-series-quad-block-pattern-mount-27-1/ |
| 1310 Series Hyper Hub (4 Start, 8mm Lead Screw Bore) | 1 | $9.99 | ServoCity: https://www.servocity.com/1310-series-hyper-hub-4-start-8mm-lead-screw-bore/ |
| 1512 Series 6mm ID Spacer (8mm OD, 8mm Length) - 4 Pack | 1 | $2.79 | ServoCity: https://www.servocity.com/1512-series-6mm-id-spacer-8mm-od-8mm-length-4-pack/ |
| 1309 Series Sonic Hub (6mm D-Bore) | 1 | $6.99 | ServoCity: https://www.servocity.co |

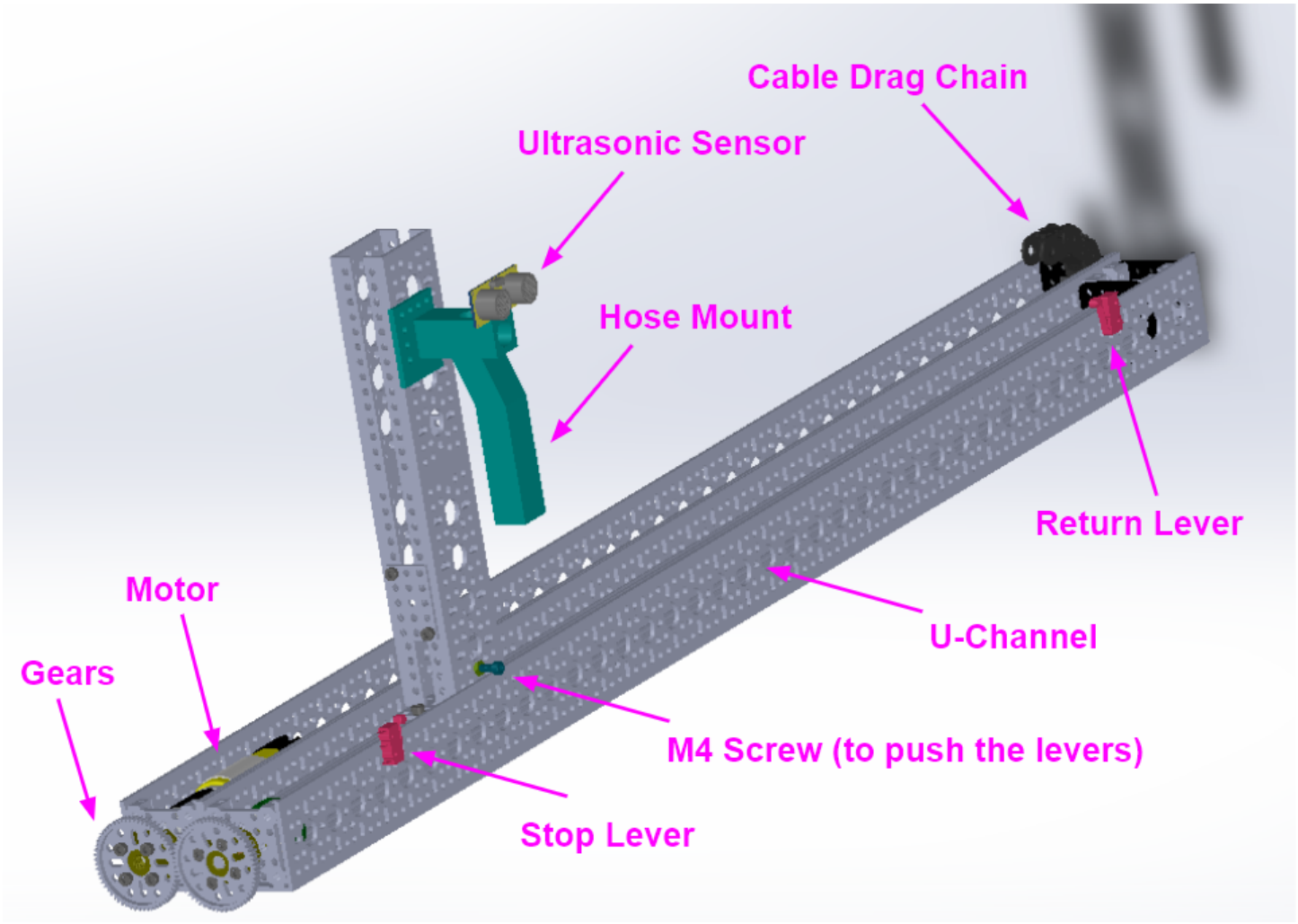
| | | | |
|---|---|---|---|
| | | | m/1309-series-sonic-hub-6mm-d-bore/ |
| 1120 Series U-Channel (37 Hole, 912mm Length) | 1 | $38.99 | ServoCity: https://www.servocity.com/1120-series-u-channel-37-hole-912mm-length/ |
| 3704 Series Plastic goRAIL Slide Plate (43-1) - 2 Pack | 2 | $2.99 | ServoCity: https://www.servocity.com/3704-series-plastic-gorail-slide-plate-43-1-2-pack/ |
| 3505 Series Lead Screw Pattern Nut (8mm Lead, 4 Start, 32mm OD, 16mm Length) | 2 | $8.99 | ServoCity: https://www.servocity.com/3505-series-lead-screw-pattern-nut-8mm-lead-4-start-32mm-od-16mm-length/ |
| 3503 Series Lead Screw Nut for Open goRAIL (8mm Lead, 4 Start, 38mm Length) | 2 | $12.99 | ServoCity: https://www.servocity.com/3503-series-lead-screw-nut-for-open-gorail-8mm-lead-4-start-38mm-length/ |
| 2807 Series Stainless Steel Shim (8mm ID x 11mm OD, 1mm Thickness) - 12 Pack | 1 | $2.59 | ServoCity: https://www.servocity.com/2807-series-stainless-steel-shim-8mm-id-x-11mm-od-1mm-thickness-12-pack/ |
| 2807 Series Stainless Steel Shim (6mm ID x 9mm OD, 0.50mm Thickness) - 12 Pack | 1 | $2.29 | ServoCity: https://www.servocity.com/2807-series-stainless-steel-shim-6mm-id-x-9mm-od-0-50mm-thickness-12-pack/ |
| 2807 Series Stainless Steel Shim (6mm ID x 9mm OD, 0.25mm Thickness) - 12 Pack | 1 | $2.29 | ServoCity: https://www.servocity.com/2807-series-stainless-steel-shim-6mm-id-x-9mm-od-0-25mm-thickness-12-pack/ |
| 3501 Series Lead Screw (8mm Lead, 4 Start, 950mm Length) | 1 | $20.99 | ServoCity: https://www.servocity.com/37-4-950mm-8mm-lead-screw/ |
| Steel Channel-Connector Plate - 2 Pack | 4 | $3.99 | ServoCity: https://www.servocity.co |

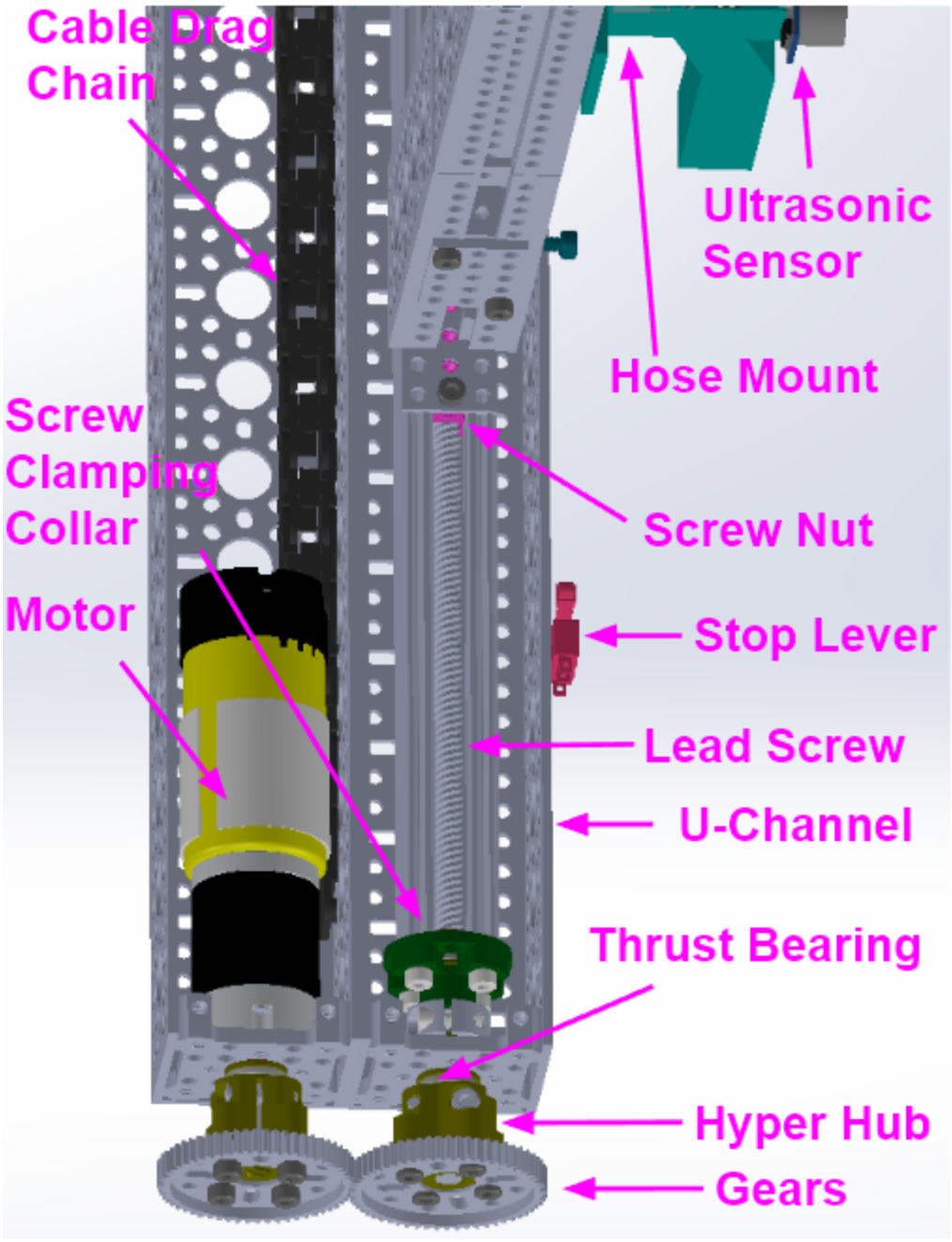
| | | | |
|---|---|---|---|
| | | | m/2803-series-stainless-steel-threaded-plate-39-2 2/ |
| 1 x 5 Steel Threaded Plate - 2 Pack | 4 | $3.99 | ServoCity: https://www.servocity.com/2803-series-stainless-steel-threaded-plate-39-7-2-pack/ |
| 2302 Series Aluminum, MOD 0.8, Hub Mount Gear (14mm Bore, 60 Tooth) | 2 | $9.99 | ServoCity: https://www.servocity.com/2302-series-aluminum-mod-0-8-hub-mount-gear-14mm-bore-60-tooth/ |
| 1611 Series Flanged Ball Bearing (8mm ID x 14mm OD, 5mm Thickness) - 2 Pack | 4 | $3.99 | ServoCity: https://www.servocity.com/1611-series-flanged-ball-bearing-8mm-id-x-14mm-od-5mm-thickness-2-pack/ |
| 1201 Series Quad Block Pattern Mount (43-2) | 2 | $6.99 | ServoCity: https://www.servocity.com/1201-series-quad-block-pattern-mount-43-2/ |
| 1118 Series Open goRAIL (912mm Length) | 1 | $20.89 | ServoCity: https://www.servocity.com/1118-series-open-gorail-912mm-length/ |
| 1116 Series Grid Plate (3 x 5 Hole, 24 x 40mm) | 2 | $1.29 | ServoCity: https://www.servocity.com/1116-series-grid-plate-3-x-5-hole-24-x-40mm/ |
| 1106 Series Square Beam (5 Hole, 40mm Length) | 4 | $2.29 | ServoCity: https://www.servocity.com/1106-series-square-beam-5-hole-40mm-length/ |
| 3504 Series Lead Screw Clamping Collar (8mm Lead, 4 Start, 21mm OD, 9mm Length) | 2 | $5.99 | ServoCity: https://www.servocity.com/3504-series-lead-screw-clamping-collar-8mm-lead-4-start-21mm-od-9mm-length/ |
| 1613 Series Thrust Ball Bearing (8mm ID x 16mm OD, 5mm Thickness) | 4 | $3.99 | ServoCity: https://www.servocity.com/1613-series-thrust-ball-bearing-8mm-id-x-16mm-od-5mm-thickness/ |
| 2800 Series Zinc-Plated Steel | 1 | $3.69 | ServoCity: |

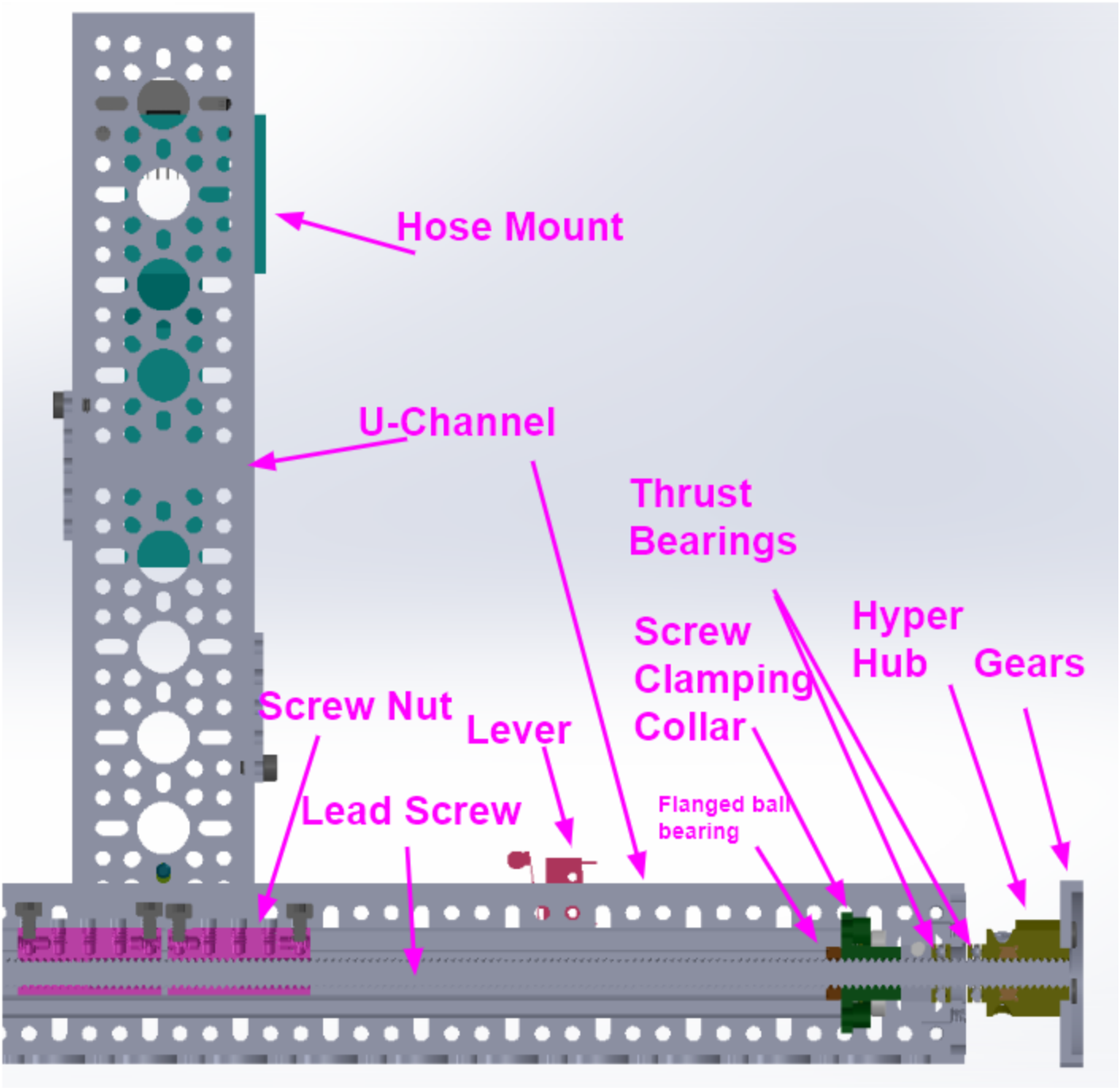
| | | | |
|---|---|---|---|
| Socket Head Screw (M4 x 0.7mm, 14mm Length) - 25 Pack | | | https://www.servocity.com/2800-series-zinc-plated-steel-socket-head-screw-m4-x-0-7mm-14mm-length-25-pack/ |
| 2800 Series Zinc-Plated Steel Socket Head Screw (M4 x 0.7mm, 11mm Length) - 25 Pack | 1 | $3.49 | ServoCity: https://www.servocity.com/2800-series-zinc-plated-steel-socket-head-screw-m4-x-0-7mm-11mm-length-25-pack/ |
| 2800 Series Zinc-Plated Steel Socket Head Screw (M4 x 0.7mm, 9mm Length) - 25 Pack | 1 | $3.29 | ServoCity: https://www.servocity.com/2800-series-zinc-plated-steel-socket-head-screw-m4-x-0-7mm-9mm-length-25-pack/ |
| 2800 Series Zinc-Plated Steel Socket Head Screw (M4 x 0.7mm, 8mm Length) - 25 Pack | 1 | $3.19 | ServoCity: https://www.servocity.com/2800-series-zinc-plated-steel-socket-head-screw-m4-x-0-7mm-8mm-length-25-pack/ |
| 2800 Series Zinc-Plated Steel Socket Head Screw (M4 x 0.7mm, 6mm Length) - 25 Pack | 1 | $2.99 | ServoCity: https://www.servocity.com/2800-series-zinc-plated-steel-socket-head-screw-m4-x-0-7mm-6mm-length-25-pack/ |
| 5202 Yellow Series Planetary Gear Motor DC | 1 | $48.98 | ServoCity: https://www.servocity.com/5202-series-yellow-jacket-planetary-gear-motor-71-2-1-ratio-84-rpm-3-3-5v-encoder/ |
| Submersible 3V DC Water Pump | 1 | $2.95 | Adafruit: https://www.adafruit.com/product/4546 |
| Red Arcade Style Button | 1 | $2.50 | ServoCity: https://www.servocity.com/spdt-miniature-limit-switch-with-roller-lever-2-pack/ |
| ALITOVE AC 100-240V to DC 12V 10A power supply | 1 | $19.99 | Amazon: https://www.amazon.com/ALITOVE-100-240V-Converter-Transformer-5-5x2-1mm/dp/B07MXXXBV8 |

| | | | |
|---|---|---|---|
| 1/4" CLear PVC Tubing | 1 | $14.99 | Amazon: https://www.amazon.com/Tubing-Flexible-Hybrid-Lightweight-10-Feet/dp/B07HF4SYWY/ref=sr_1_2_sspa?keywords=plastic+tubing+1%2F4&qid=1670828020&sr=8-2-spons&psc=1&spLa=ZW5jcnlwdGVkUXVhbGlmaWVyPUEzQUlaMk5VNzI3V0szJmVuY3J5cHRlZElkPUEwNTQ0NjE4MzlQOVlUQTY1RElVVSZlbmNyeXB0ZWRBZElkPUEwMjE4MzAwRkpGU0VSUVlOWU1RJndpZGdldE5hbWU9c3BfYXRmJmFjdGlvbj1jbGlja1JlZGlyZWN0JmRvTm90TG9nQ2xpY2s9dHJ1ZQ== |

***Images of the CAD***:




Cable Drag Chain
Ultrasonic Sensor
Hose Mount
Screw Clamping Collar
Screw Nut
Motor
Stop Lever
Lead Screw
U-Channel
Thrust Bearing
Hyper Hub
Gears


Hose Mount
U-Channel
Thrust Bearings
Hyper Hub
Gears
Screw Clamping Collar
Screw Nut
Lever
Lead Screw
Flanged ball bearing

***Entire Code***:

```
#include <ESP32Encoder.h>
#include <IRremote.h>
#include <ezButton.h>

#define LED_PIN 13//built in LED pin
#define IR 15     //IR receiver pin
#define PMP 12    //water pump pin
#define TRIG 18  //ultrasonic transmitter pin
#define ECHO 19   //ultrasonic receiver pin
#define POT 32   //potentiometer
#define RPWM 4 // define pin 3 for RPWM pin (output)
#define R_EN 26
#define LPWM 16 // define pin 6 for LPWM pin (output)
#define L_EN 25
ezButton SW1(27);
ezButton SW2(33);


ESP32Encoder encoder;

//Setup timer to measure watering time ----------------------------
hw_timer_t * timer = NULL;
hw_timer_t * timer1 = NULL;
uint64_t time_val;
int start_time_delay = 0;
int start_time_disable = 0;

//Setup IR receiver ---------------------------
IRrecv irrecv(IR);
decode_results results;
unsigned long last_signal = 0;


// Setting PWM properties ----------------------------
const int freq = 5000;
const int ledChannel_1 = 1;
const int ledChannel_2 = 2;
const int resolution = 8;
const int MAX_PWM_VOLTAGE = 255;
const int NOM_PWM_VOLTAGE = 150;
int D = 0;
```

```
const int max_dist = -5000; //counts by encoder, when hit this point, turn around
const int min_dist = 0; //when hit this point, stop

//Setup ultrasonic sensor ----------------------------
const float sound_speed = 0.034;
long duration = 0;
float distance = 0; //cm
int count = 0;

//Setup states ---------------------------
int state = 0;
volatile bool finished_watering = false;
volatile bool disable_sensor = false;
volatile bool delay_sensor = false;

//Initialization -----------------------------------
void setup() {
  pinMode(IR, INPUT); //IR receiver
  pinMode(ECHO, INPUT); //Ultrasonic
  pinMode(POT, INPUT); //potentiometer

  pinMode(LED_PIN, OUTPUT); //built in LED
  pinMode(PMP, OUTPUT); //water pump
  pinMode(TRIG, OUTPUT); //ultrasonic
  pinMode(RPWM, OUTPUT);
  pinMode(LPWM, OUTPUT);
  pinMode(R_EN, OUTPUT);
  pinMode(L_EN, OUTPUT);

  digitalWrite(LED_PIN, LOW); // sets the initial state of LED as turned-off
  irrecv.enableIRIn(); //configure IR
  timer = timerBegin(0, 80, true); //timer for water pump
  timer1 = timerBegin(0, 80, true);
  SW1.setDebounceTime(50); // set debounce time to 50 milliseconds
  SW2.setDebounceTime(50); // set debounce time to 50 milliseconds

  Serial.begin(115200);
}


void loop() {
  SW1.loop(); // MUST call the loop() function first
  SW2.loop();
  int front_end = SW1.getState();
  int back_end = SW2.getState();
  //Serial.println(gate);

  //Process potentiometer input to get motor pwm
  D = map(analogRead(POT), 0, 4095, 0, 100); //in percent

  //Process IR receiver input to change state if applicable
  if (irrecv.decode(&results)){
    Serial.print("Signal: ");
    Serial.println(results.value, HEX);
    if (newinput(results.value)) {
      analyze_input(results.value);
    }
    irrecv.resume();
  }

  //State machine
  switch (state) {
    case 0: //unarmed state
      light_off();
      stop_motor();
      timerStop(timer);
      timerRestart(timer);
      Serial.println("State 0: Idle");
      break;
```

```
case 1: //moving forward
  light_on();
  timerStop(timer);
  timerRestart(timer);
  finished_watering =false;
  Serial.println(back_end);
  if (back_end == HIGH) {
    //Serial.println("Moving forward");
    //move_forward(D);
    rotate(D, 0);
    if (disable_sensor == true) {
      temp_disable_ultrasonic(); //if hit an object, trigger state 3
    } else {
      check_sensor();
      Serial.println("Using sensor");
    }
  } else {
    Serial.println("SW2 triggered");
    stop_motor();
    state = 2; //turn around, moving backward
  }
  break;

case 2: //moving backward
  light_on();
  stop_motor();
  timerStop(timer);
  timerRestart(timer);
  Serial.println("Moving back");
  if (front_end == HIGH) {
    //move_backward(D);
    rotate(D, 1);
  } else {
    state = 0; //finished cycle
  }
  break;
```

```
    case 3: //Watering when encounter the object
      light_on();
      if (delay_sensor == true) {
         temp_delay_sensor();
      } else {
        countDownWater();
        if (finished_watering ==true) {
          Serial.println("Transision to state 1");
          disable_sensor = true;
          state = 1;
        }
      }
      break;
    }
}

bool newinput(unsigned long value) {
  if (value != last_signal) {
    if ((value == 0xFF02FD) or (value == 0xFFC23D) or (value == 0xFF22DD)) {
      last_signal = value;
      return true;
    } else {
      return false;
    }
  }
}

void analyze_input(unsigned long value) {
  switch (value) {
    case 0xFF02FD:
      Serial.println("Changed to state 0 from remote");
      state = 0;
      break;
    case 0xFFC23D:
      state = 1;
      Serial.println("Changed to state 1 from remote");
      break;
    case 0xFF22DD:
      state = 2;
      Serial.println("Changed to state 2 from remote");
      break;
  }
```

```
void light_off() {
  digitalWrite(LED_PIN, LOW);
}

void light_on() {
  digitalWrite(LED_PIN, HIGH);
}

void move_forward(int D) {
  analogWrite(LPWM, 0);
  analogWrite(RPWM, D);
}

void move_backward(int D) {
  analogWrite(RPWM, 0);
  analogWrite(LPWM, 255);
}

void check_sensor() {

  if (current_dist() != 0) {
    stop_motor();
    state = 3;
  }
}


int current_dist() {
  digitalWrite(TRIG, LOW); //set trigger signal low for 2us
  delayMicroseconds(2);

   /*send 10 microsecond pulse to trigger pin of HC-SR04 */
  digitalWrite(TRIG, HIGH);  // make trigger pin active high
  delayMicroseconds(10);            // wait for 10 microseconds
  digitalWrite(TRIG, LOW);   // make trigger pin active low

  /*Measure the Echo output signal duration or pulss width */
  duration = pulseIn(ECHO, HIGH); // save time duration value in "duration variable
  distance= duration*0.034/2; //Convert pulse duration into distance
  if (distance > 10) { //limit sensor range to 20cm so not mistaken other objects
    distance = 0;
  }
  if (distance != 0) {
    if (count < 900) {
      count = count + 1;
      Serial.print("Count: ");
      Serial.print(count);
      return 0;
    } else {
      count = 0;
    }
  }
  Serial.print("Distance: ");
  Serial.println(distance);
  return distance;
}
```

```
void countDownWater() {
  if (!timerStarted(timer)) {
    timerStart(timer);
    digitalWrite(PMP, HIGH);
    Serial.println("Starting watering");
  } else {
    time_val = timerReadSeconds(timer);
    Serial.print("Watering time (s): ");
    Serial.println(time_val);
    if (time_val > 5) {
      timerStop(timer);
      timerRestart(timer);
      finished_watering =true;
      stop_motor();
      count = 0;
      Serial.println("Finished watering" );
    }
  }
}


void rotate(int value, int dir) {

  digitalWrite(L_EN, HIGH);
  digitalWrite(R_EN, HIGH);
  int pwm1Pin, pwm2Pin;
  if(dir == 1){
    analogWrite(LPWM, toPWM(value));
    analogWrite(RPWM, 0);
  }else{
    analogWrite(LPWM, 0);
    analogWrite(RPWM, toPWM(value));
  }
  digitalWrite(pwm1Pin, LOW);
  if(value >=0 && value <=100 ){
    analogWrite(pwm2Pin, toPWM(value));
  }
}
```

```
void stop_motor(){
  digitalWrite(L_EN, LOW);
  digitalWrite(R_EN, LOW);
  digitalWrite(PMP, LOW);
  //Serial.println("Motor stopped");
}

int toPWM(int v){
  return map(v, 0, 100,0,255);
}

void temp_disable_ultrasonic() {
  int current_time = millis();
  if (start_time_disable == 0) {
    start_time_disable = current_time;
  } else {
    int time_diff = current_time - start_time_disable;
    Serial.println("Pausing time: ");
    Serial.print(time_diff);
    if (time_diff >= 12000) {
      check_sensor();
      start_time_disable = 0;
      disable_sensor = false;
      Serial.println("Stop pausing");
      count == 0;
    }
  }
}

void temp_delay_sensor() {
  int current_time = millis();
  if (start_time_delay == 0) {
    start_time_delay = current_time;
  } else {
    int time_diff = current_time - start_time_delay;
    Serial.println("Delaying time: ");
    Serial.print(time_diff);
    if (time_diff >= 5000) {
      check_sensor();
      start_time_delay = 0;
      delay_sensor = false;
      Serial.println("Stop delaying");
    }
  }
}
```