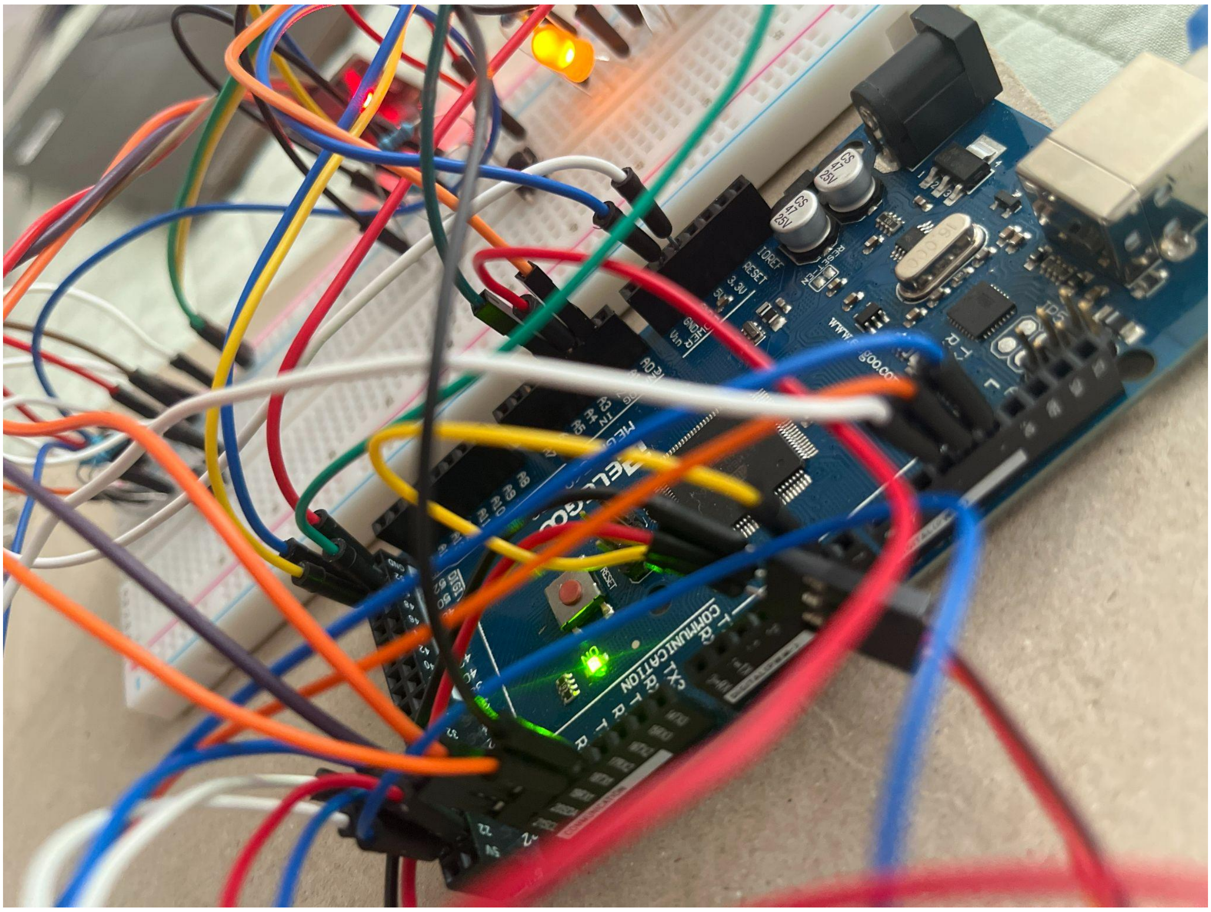
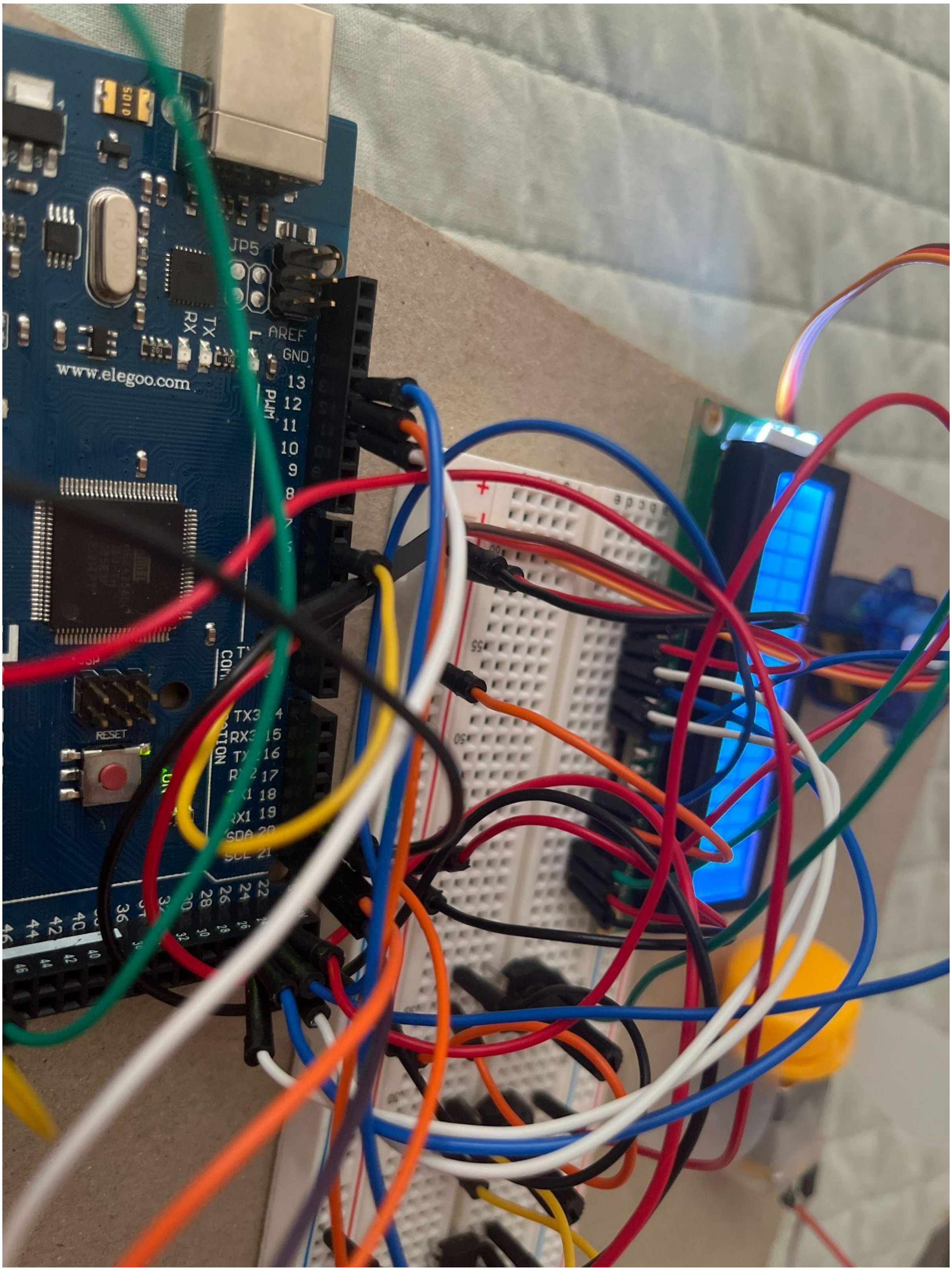
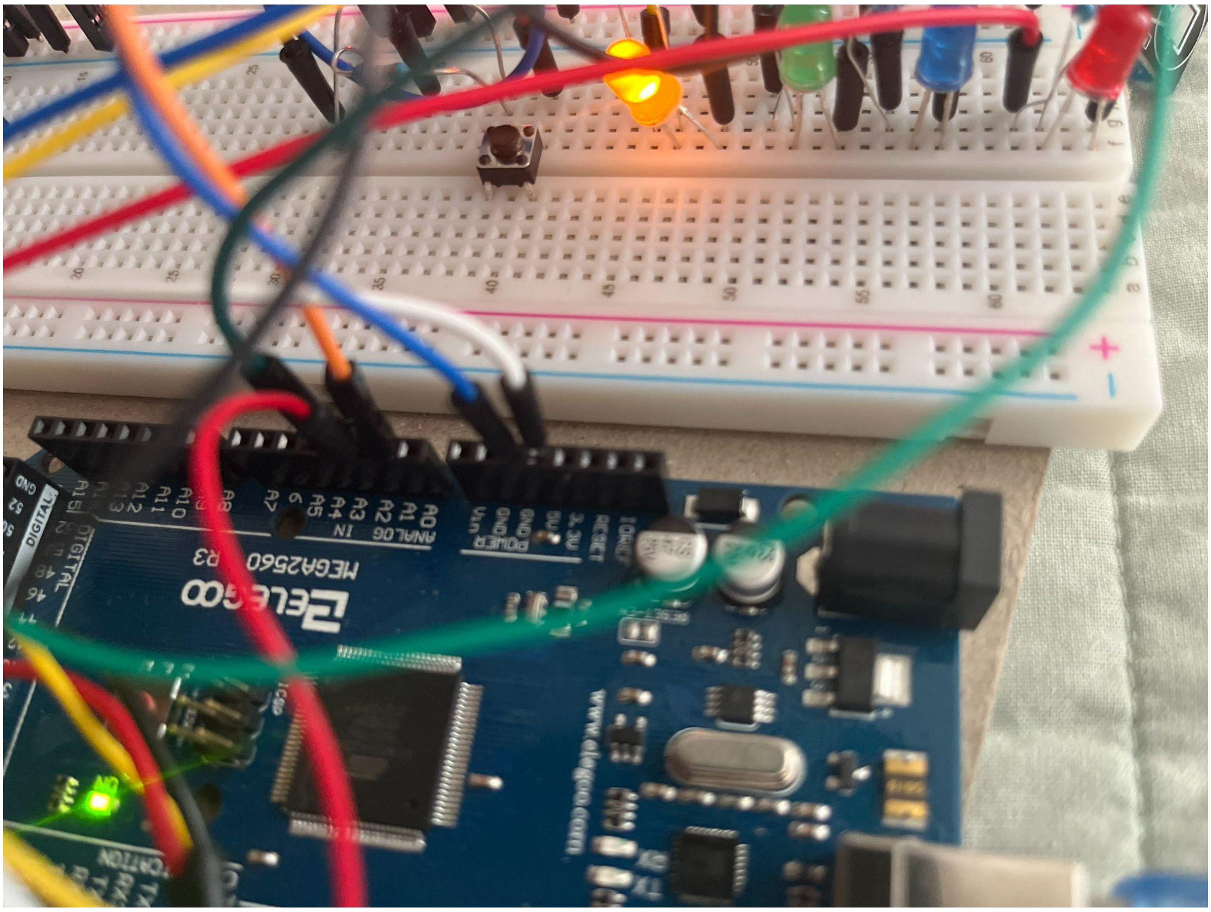
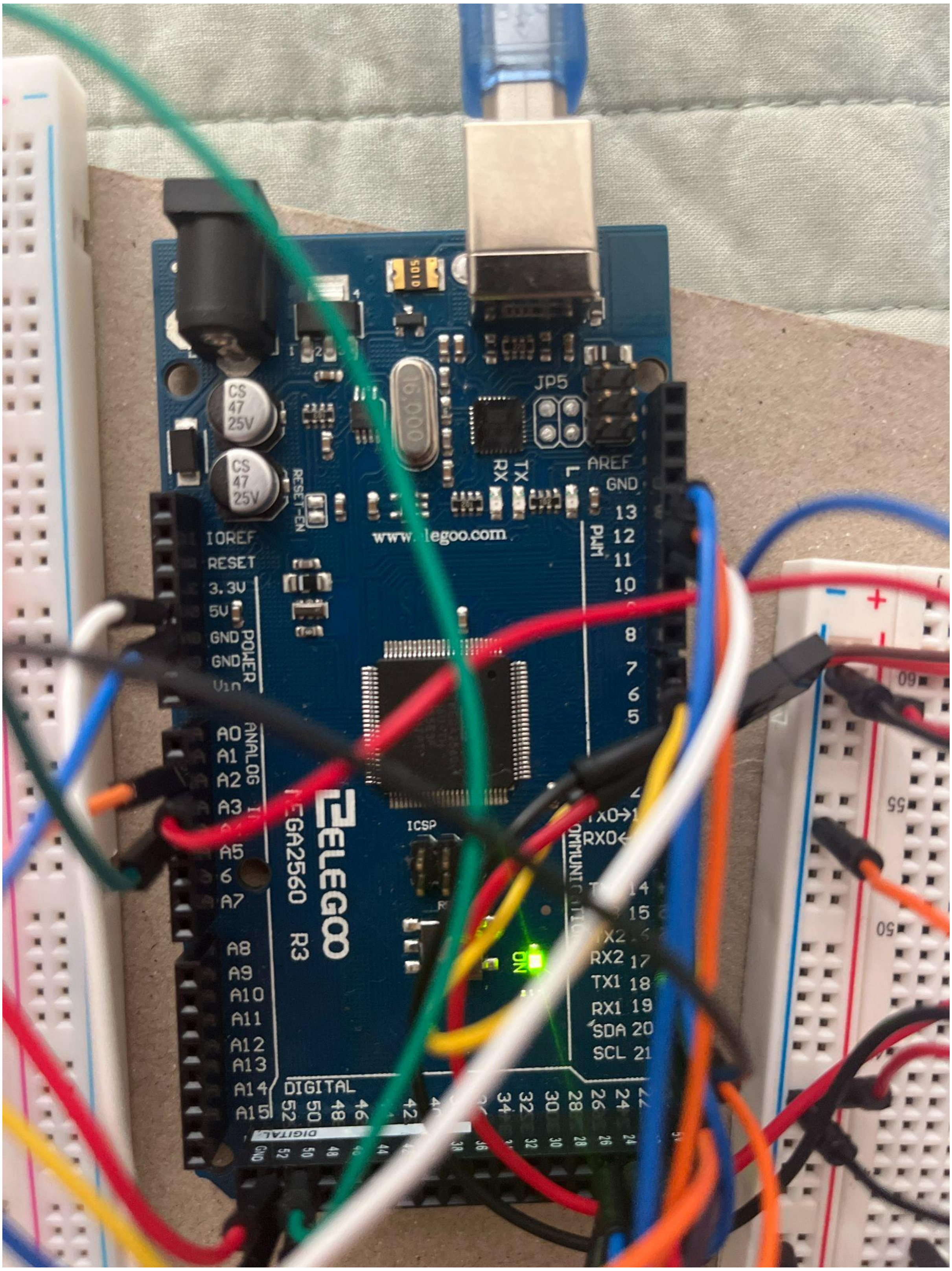
```
#include "DHT.h"
#include <LiquidCrystal.h>
#include <Servo.h>

#define ENABLE 13
#define DIRA 12
#define DIRB 11


// Include required libraries
#include <Wire.h>
#include <TimeLib.h>
#include <DS1307RTC.h>

int state = LOW;      // the current state of the output pin
int reading;    // the current reading from the input pin
int previous = HIGH;    // the previous reading from the input pin

int reset;

int printed_disabled = false;
int printed_idle = false;
int printed_running  = false;
int printed_error = false;


// Define constant for RTC I2C Address
#define DS1307_CTRL_ID 0x68

float Temp_Threshold = 75;
float Water_Threshold = 250;

#define DHTPIN A3
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

Servo myservo;
int val;
int button = 0;
LiquidCrystal lcd(22, 23, 24, 25, 26, 27);

volatile unsigned char* my_ADMUX = (unsigned char*) 0x7C;
volatile unsigned char* my_ADCSRB = (unsigned char*) 0x7B;
volatile unsigned char* my_ADCSRA = (unsigned char*) 0x7A;
volatile unsigned int* my_ADC_DATA = (unsigned int*) 0x78;

// B register
```

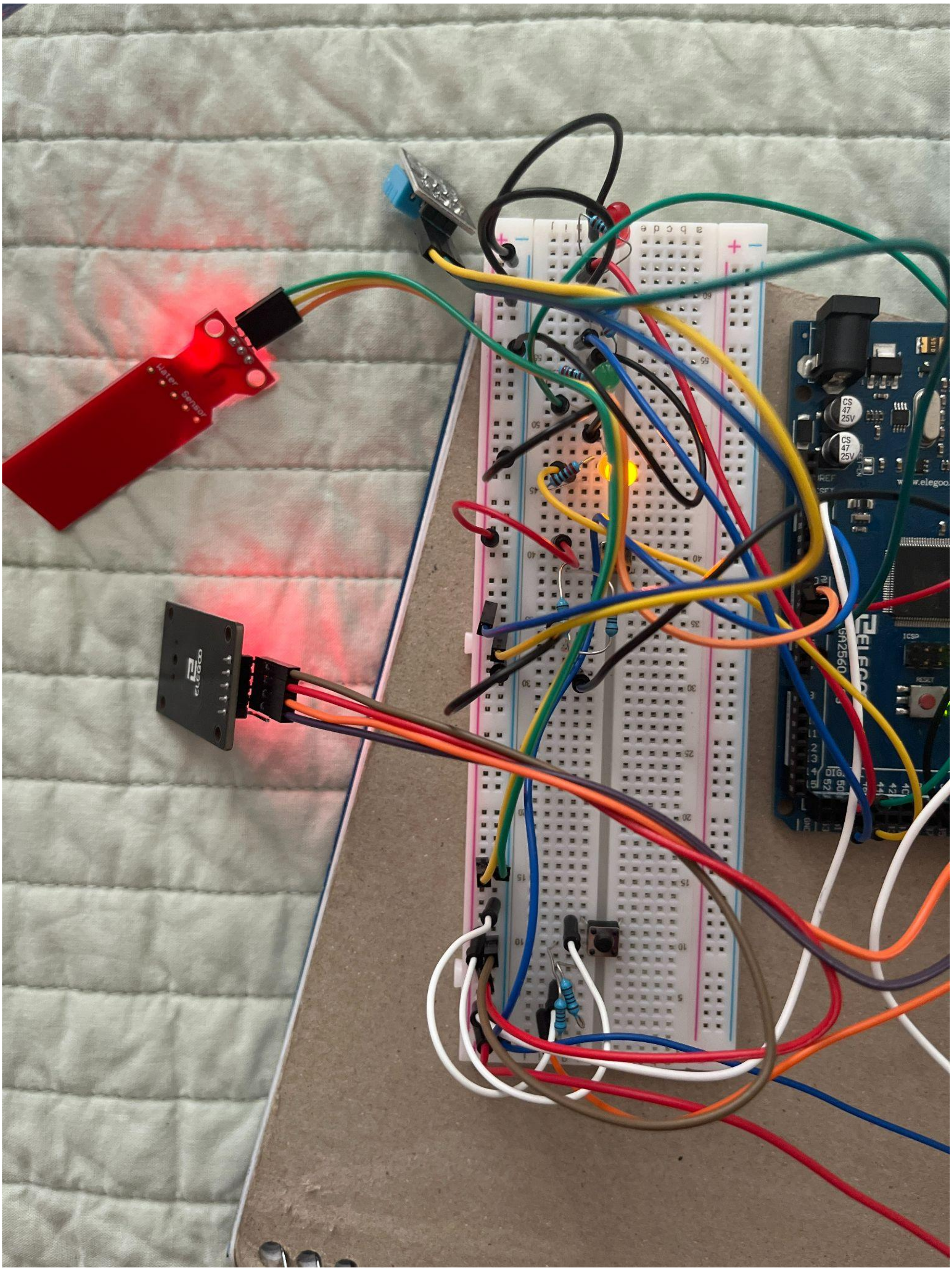
```
}

volatile unsigned char* port_b = (unsigned char*) 0x25; // Setting the port_b (data register) to address 0x25 (sets bit as high or low, outputs data)
volatile unsigned char* ddr_b = (unsigned char*) 0x24;  // Setting the ddr_b (Data Direction Register) to address 0x24 (sets it as input or output)
volatile unsigned char* pin_b = (unsigned char*) 0x23;  // Setting pin_b (Input Pin Address) to 0x23 (Reading a value from a pin)

// K register
volatile unsigned char* pin_k = (unsigned char*) 0x106;
volatile unsigned char* ddr_k = (unsigned char*) 0x107;
volatile unsigned char* port_k = (unsigned char*) 0x108;

//D register
volatile unsigned char* pin_d = (unsigned char*) 0x09;
volatile unsigned char* ddr_d = (unsigned char*) 0x0A;
volatile unsigned char* port_d = (unsigned char*) 0x0B;

long time = 0;         // the last time the output pin was toggled
long debounce = 200;   // the debounce time, increase if the output flickers

void setup() {
    lcd.begin(16, 2);
    adc_init(); // setup the ADC
    dht.begin();
      myservo.attach(6);
    *ddr_b = B11101111;

    *ddr_d = B11110010;

    *ddr_k = B11111110;
    attachInterrupt(digitalPinToInterrupt(18),handleInt,CHANGE);
    attachInterrupt(digitalPinToInterrupt(19),handleInt2,CHANGE);
    Serial.begin(9600);
}

void loop() {
  if (reading == HIGH && previous == LOW && millis() - time > debounce) {
      if (state == HIGH){
      state = LOW;
      Serial.print("OFF\n");
    }
    else{
      Serial.print("ON\n");
      state = HIGH;
     }
    time = millis();
   }
  previous = reading;
  reading = LOW;

   if(state == LOW){
      disabled_state();
     }

   else if(state == HIGH){
     if (Water_level() > Water_Threshold && DHT_sensor() < Temp_Threshold){

      idle_state();
     }
     else if (Water_level() > Water_Threshold && DHT_sensor() > Temp_Threshold){

      running_state();
     }
     else if(Water_level() < Water_Threshold){

      error_state();
     }
   }


}

// Interrupt Handler
void handleInt() {
  reading = HIGH;
}

void handleInt2() {
  reset = HIGH;
}

void disabled_state(){

   printed_idle = false;
   printed_running  = false;
   printed_error = false;
  if (printed_disabled == false){
  Serial.println("DISABLED State");
  printCurrentTime();
  printed_disabled = true;
  }

  *port_b &= B00000100;
  *port_b |= B00000100;
  lcd.clear();
}

void idle_state(){
  printed_disabled = false;

printed_running  = false;
 printed_error = false;
    if (printed_idle == false){
  Serial.println("IDLE State");
  printCurrentTime();
  printed_idle = true;
  }

  *port_b &= B00001000;
  *port_b |= B00001000;

   DHT_sensor();
   servo_motor();



}

void running_state(){
   printed_disabled = false;

 printed_idle  = false;
 printed_error = false;
    if (printed_running == false){
  Serial.println("RUNNING State");
  printCurrentTime();
  printed_running =  true;
  }


  DHT_sensor();
  *port_b &= B11100001;
  *port_b |= B00000001;
  fan_control();
  servo_motor();


}

void error_state(){
   printed_disabled = false;
 printed_running  = false;
 printed_idle = false;
   LCD_error();
   if(printed_error == false){
    Serial.println("ERROR State");
  printCurrentTime();
  printed_error =  true;
  }

  *port_b &= B00000010;
  *port_b |= B00000010;



  if(Water_level() > Water_Threshold && reset  == HIGH){
    idle_state();
    reset = LOW;
  }

}

void LCD_error()
{
  lcd.clear();
  lcd.setCursor (0,0);
  lcd.print("     ERROR");

  lcd.setCursor(0,1);
  lcd.print ("   LOW WATER");
}
float DHT_sensor(){
  float h = dht.readHumidity(); // Read humidity
  float f = dht.readTemperature(true);
   if (isnan(h) || isnan(f)) // Check if any reads failed and exit early (to try again).
  {
    Serial.println(F("Failed to read from DHT sensor!"));
  }
  LCD_data(h, f);
  return f;
}

void LCD_data(float h, float f)
{
  lcd.setCursor (0,0);
  lcd.print ("Humidity: ");
  lcd.print (h);
  lcd.print ("%");
  lcd.setCursor (0,1);
  lcd.print ("Temp: ");
  lcd.print (f);
  lcd.print (" F");
}

void fan_control(){
 *port_b |= B11110000;

  }

void servo_motor (){
  val = adc_read(2);
  val= map(val , 0 , 1023, 0 , 180);
  myservo.write(val);

  delay(100);
}

double Water_level()
{
  // get the reading from the ADC
  unsigned int adc_reading = adc_read(1);
  return adc_reading;
}


// Format numbers as 2-digit numbers
void print2digits(int number) {
  if (number >= 0 && number < 10) {
    Serial.print('0');
  }
  Serial.print(number);
}

// Print to the serial monitor
void printCurrentTime(){

tmElements_t tm;

  if (RTC.read(tm)) {
    print2digits(tm.Hour);
    Serial.print(':');
    print2digits(tm.Minute);
    Serial.print(':');
    print2digits(tm.Second);
    Serial.print(" - ");
    Serial.print(tmYearToCalendar(tm.Year));
    Serial.print('-');
    print2digits(tm.Month);
    Serial.print('-');
    print2digits(tm.Day);

    Serial.println();
    delay(500);
  }
}

void adc_init()
{
  // setup the A register
  // set bit   7 to 1 to enable the ADC
  *my_ADCSRA |= 0b10000000;
  // clear bit 5 to 0 to disable the ADC trigger mode
  *my_ADCSRA &= 0b11011111;
  // clear bit 4 to 0 to disable the ADC interrupt
  *my_ADCSRA &= 0b11110111;
  // clear bit 3-0 to 0 to set prescaler selection to slow reading
  *my_ADCSRA &= 0b11111000;


  // setup the B register
  // clear bit 3 to 0 to reset the channel and gain bits
  *my_ADCSRB &= 0b11110111;
  // clear bit 2-0 to 0 to set free running mode
  *my_ADCSRB &= 0b11111000;


  // setup the MUX Register
  // clear bit 7 to 0 for AVCC analog reference
  *my_ADMUX &= 0b01111111;
  // set bit   6 to 1 for AVCC analog reference
  *my_ADMUX |= 0b01000000;
  // clear bit 5 to 0 for right adjust result
  *my_ADMUX &= 0b11011111;
    // clear bit 5 to 0 for right adjust result
  *my_ADMUX &= 0b11011111;
  // clear bit 4-0 to 0 to reset the channel and gain bits
  *my_ADMUX &= 0b11100000;


}
unsigned int adc_read(unsigned char adc_channel_num)
{
  // clear the channel selection bits (MUX 4:0)
  *my_ADMUX &=0b11100000;

  // clear the channel selection bits (MUX 5)
  *my_ADCSRB &= 0b11110111;

  // set the channel selection bits, but remove the most significant bit (bit 3)
  if(adc_channel_num>7){
    adc_channel_num -=8;
    // set MUX bit 5
    *my_ADCSRB |= 0b00001000;
  }

  // set the channel selection bits
  *my_ADMUX += adc_channel_num;

  // set bit 6 of ADCSRA to 1 to start a conversion
  *my_ADCSRA |=0x40;

  // wait for the conversion to complete
  while((*my_ADCSRA & 0x40)!=0);

  // return the result in the ADC data register
  return *my_ADC_DATA;
}
```

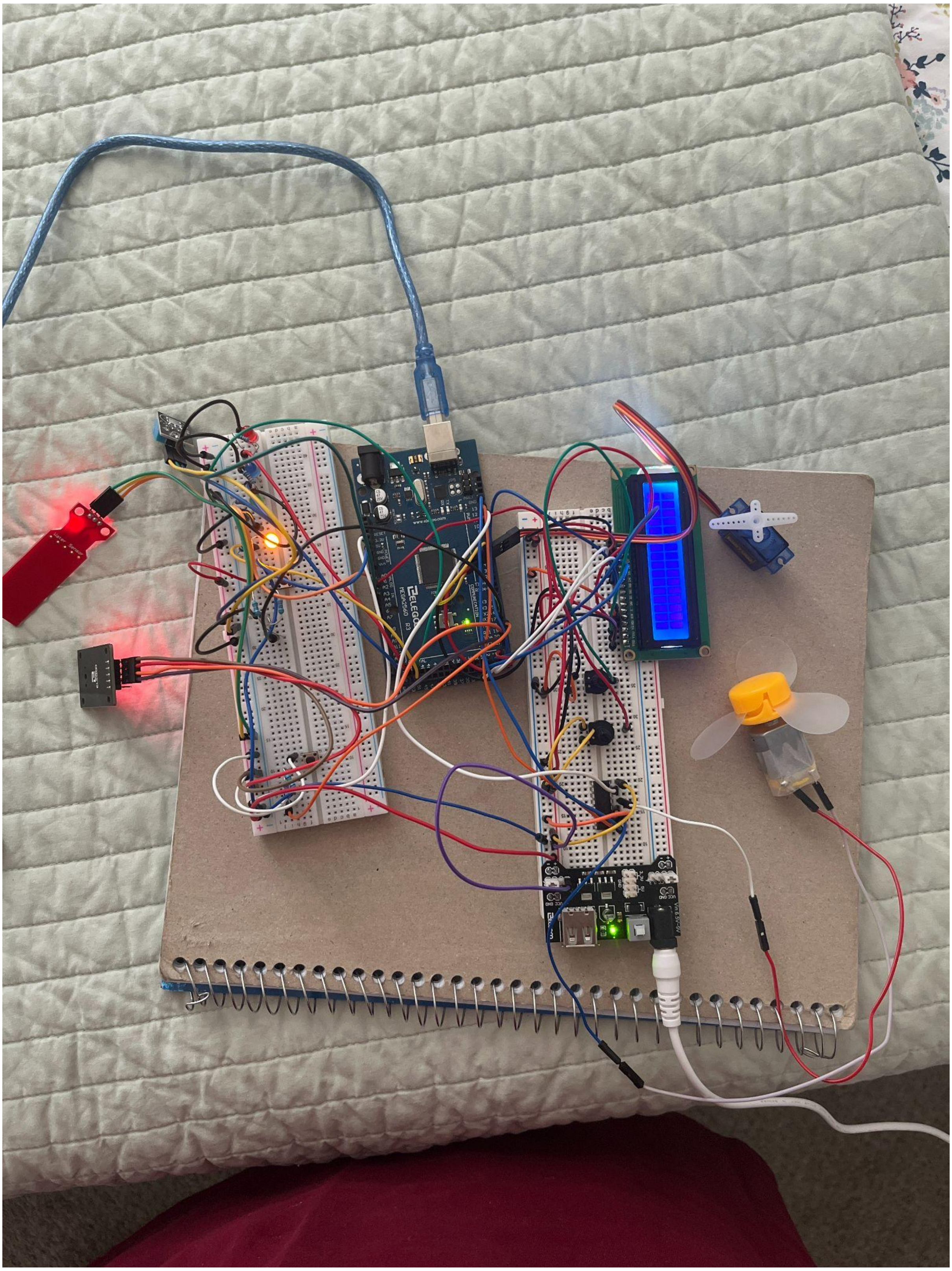

COMMUNICATION
RESET
IOREF
3.3V
5V
Vin
16.000

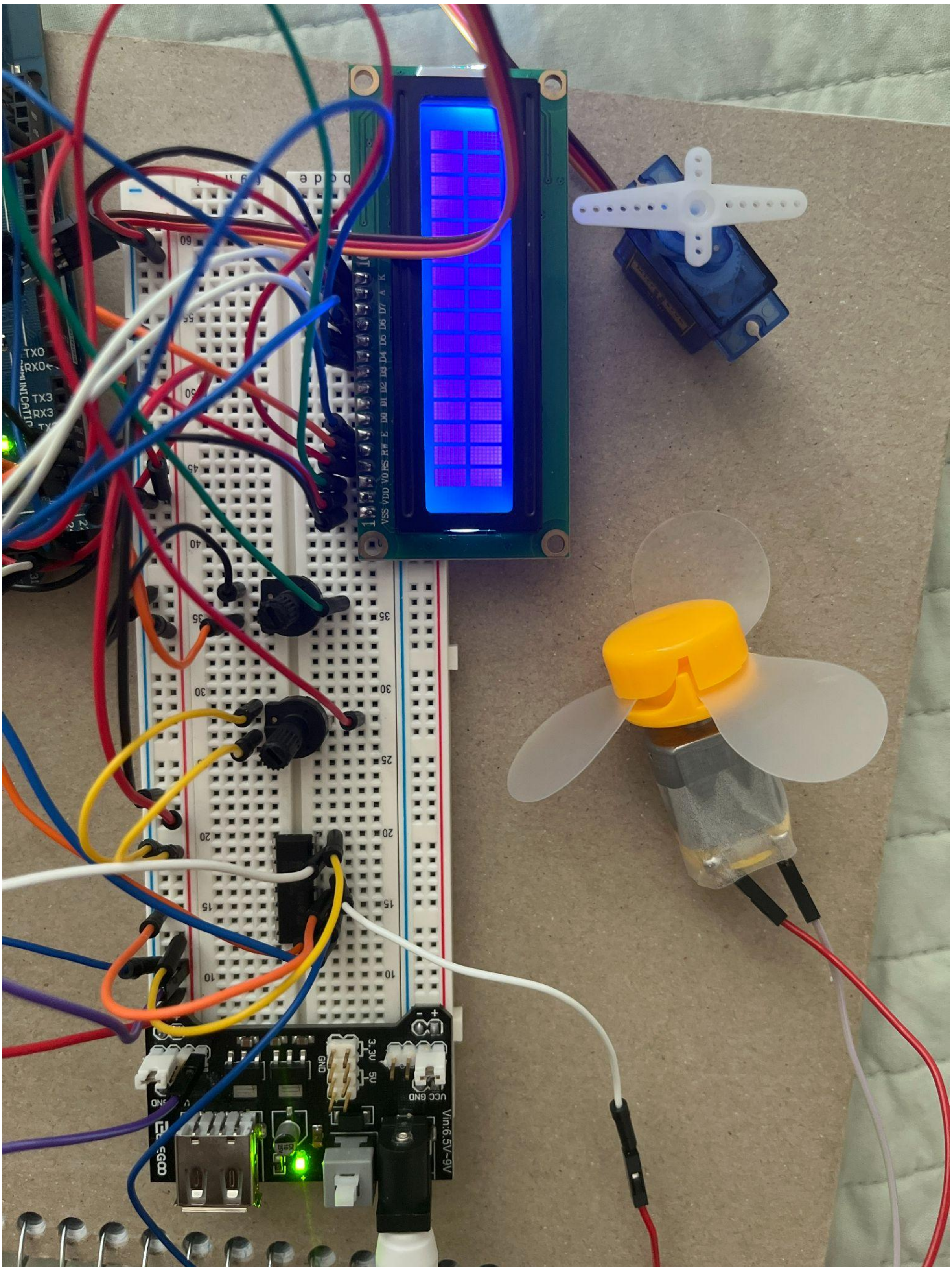

www.elegoo.com
AREF
GND
PWM
RESET
TX3 14
RX3 15
TX2 16
RX2 17
TX1 18
RX1 19
SDA 20
SCL 21


ELEGOO
MEGA2560 R3
www.elegoo.com


ELEGOO
MEGA2560 R3
www.elegoo.com
IOREF
RESET
3.3V
5V
GND
GND
Vin
POWER
ANALOG
A0
A1
A2
A3
A5
A7
A8
A9
A10
A11
A12
A13
A14
A15
AREF
GND
PWM
13
12
11
10
8
7
6
5
COMMUNICATION
TX0→1
RX0←
RX2 17
TX1 18
RX1 19
SDA 20
SCL 21
DIGITAL
ICSP
JP5
RX
TX
L
ON
16.000
CS 47 25V
RESET-EN


Water Sensor
ELEGOO
www.elegoo.
CS 47 25V
CS 47 25V
ICSP
RESET
GA2560
ELEGO


ELEGOO
MEGA2560


VSS VDD V0 RS RW E D0 D1 D2 D3 D4 D5 D6 D7 A K
TX0
RX0
TX3
RX3
3.3V
5V
GND
VCC GND
Vin:6.5V~9V