ANNEXURE I

Arduino Uno, Nano and Mega 2560 microcontroller development boards

AI.1 Arduino Uno Rev 3 microcontroller board

Arduino Uno Rev 3 is an open-source microcontroller board based on Microchip ATmega328p (8-bit AVR core) microcontroller primarily developed by Arduino.cc in the year 2010. The key features of Uno Rev 3 are: Clock speed of 16 MHz, Flash Memory: 32 KB, SRAM: 2 KB, EEPROM: 1 KB, USART peripherals: 1, SPI peripherals: 1, I²C peripherals: 1, Operating Voltage: 5 Volts, Digital I/O Pins: 14, PWM Pins: 6, Analog Input Pins: 6, DC Current per I/O Pin: 20 mA, DC Current for 3.3V Pin: 50 mA, Length: 68.6 mm, Width: 53.4 mm, Mass: 25 g.

Detailed pin configurations collected from the official Arduino.cc website are given in the Figure A1.1-4.

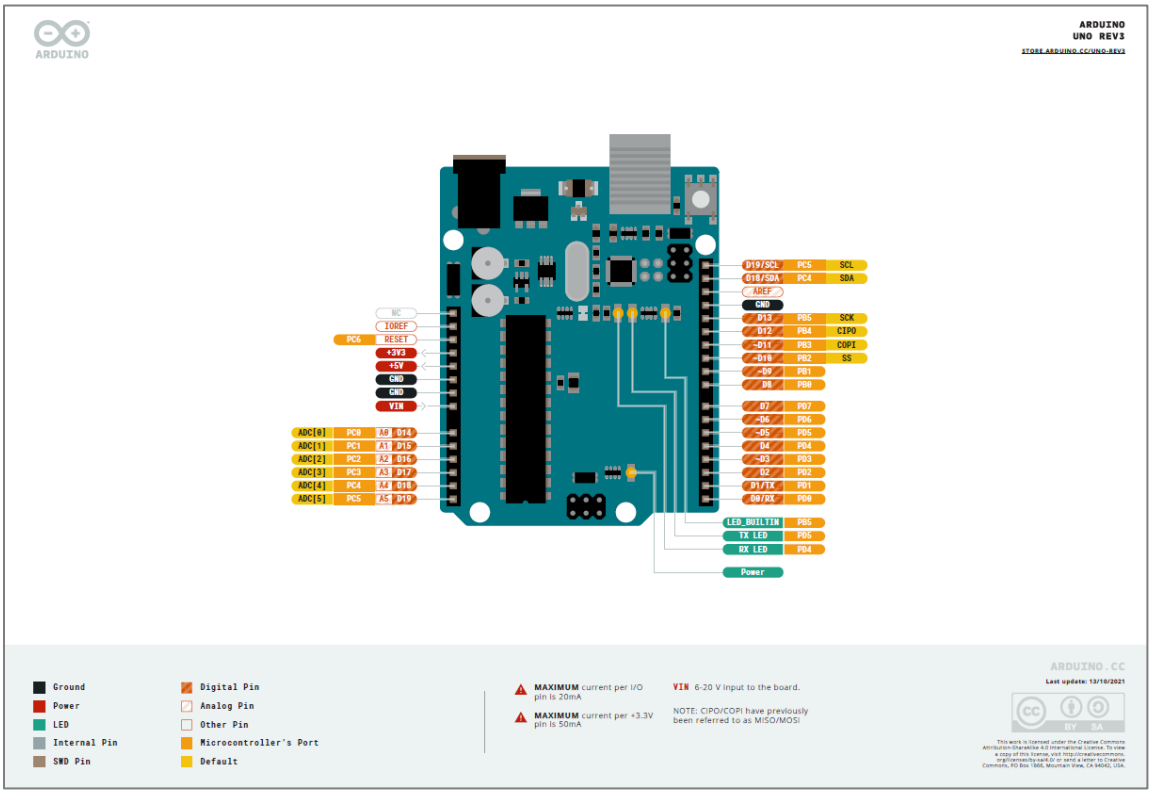

Figure A1.1: Pin diagram of Arduino Uno Rev 3 microcontroller Part I

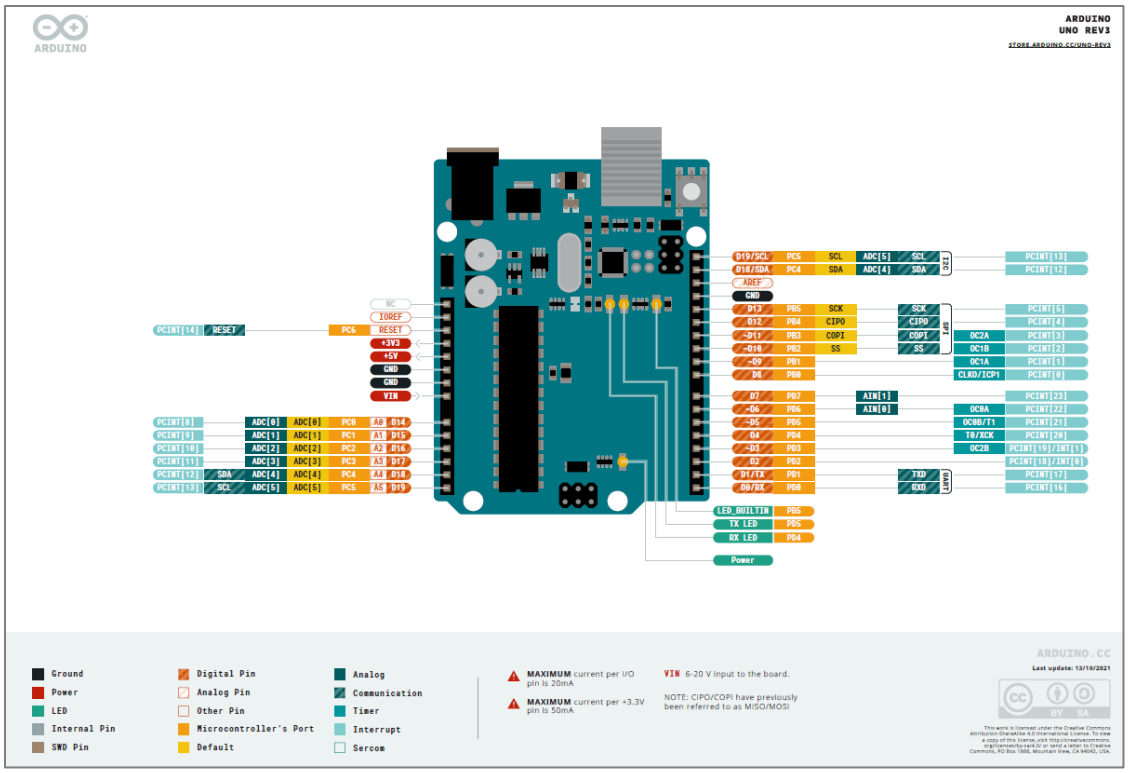

Figure A1.2: Pin diagram of Arduino Uno Rev 3 microcontroller Part II

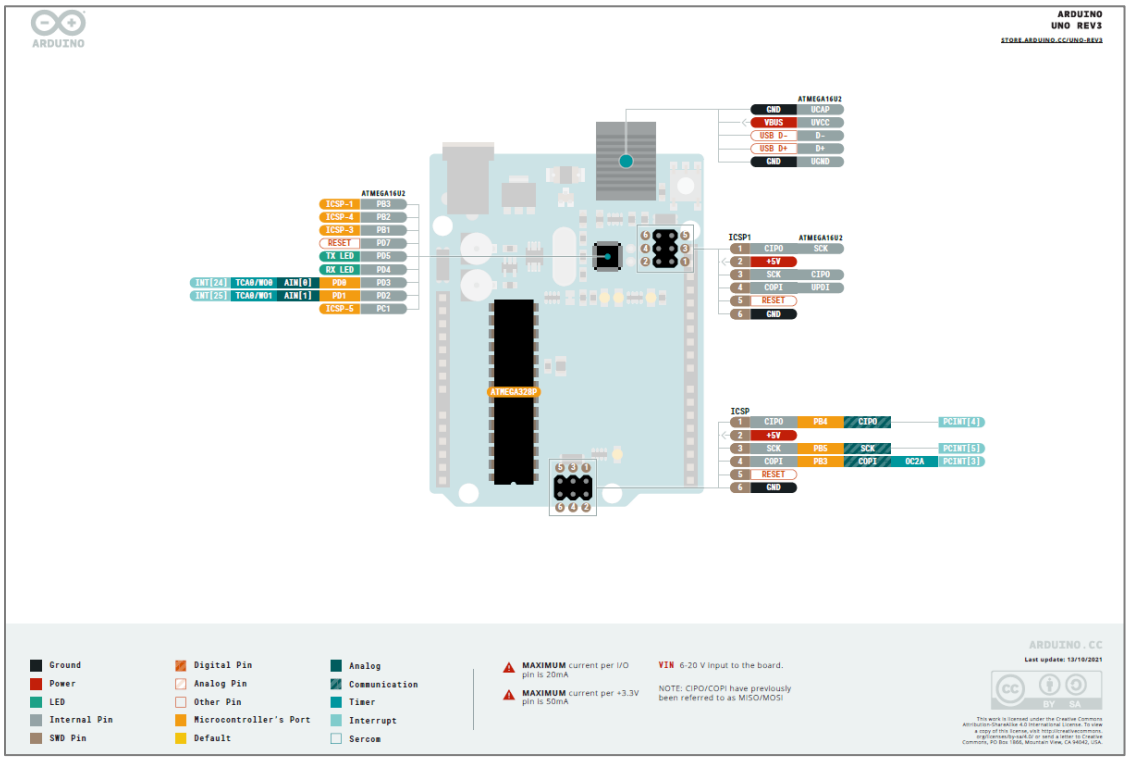

Figure A1.3: Pin diagram of Arduino Uno Rev 3 microcontroller Part III

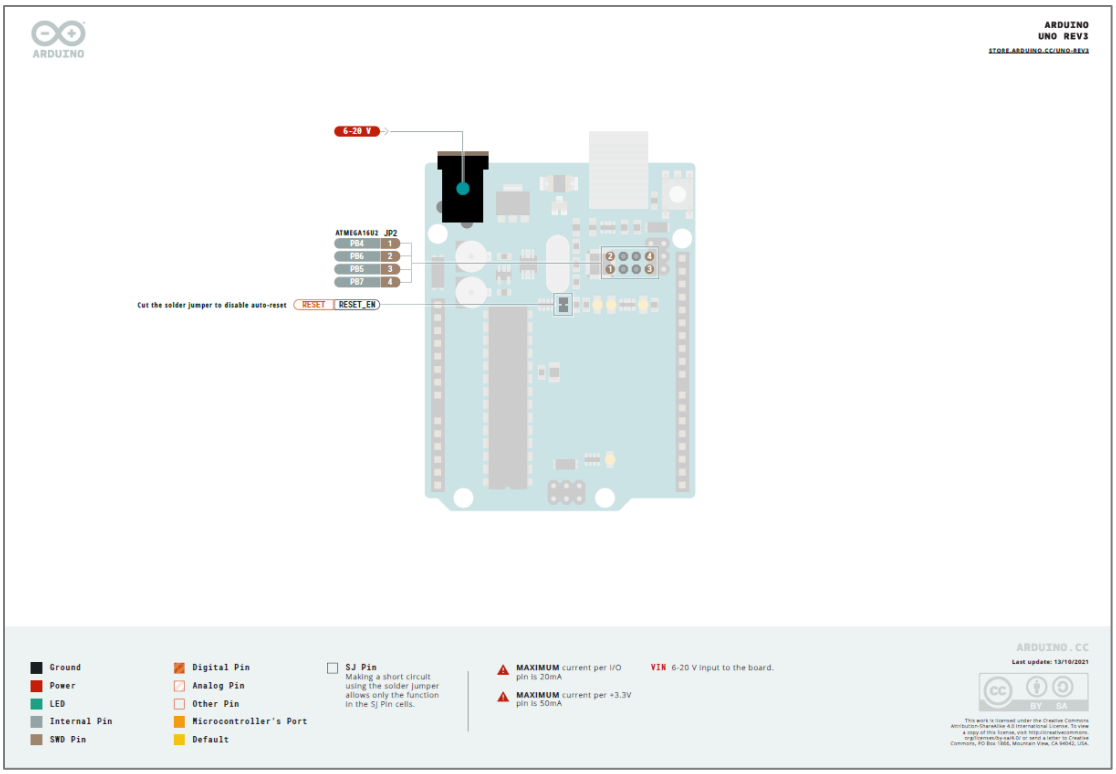

Figure A1.4: Pin diagram of Arduino Uno Rev 3 microcontroller Part IV

AI.2 Arduino Nano microcontroller board

Arduino Nano is an open-source microcontroller board based on Microchip ATmega328p microcontroller, primarily developed by Arduino.cc and initially released in 2008. The key features of Arduino Nano are: Operating voltage: 5 volts, Input voltage: 5 to 20 volts, Digital I/O pins: 14, PWM outputs: 6, Analog input pins: 8, DC per I/O pin: 40 mA, DC for 3.3 V pin: 50 mA, Flash memory: 32 KB, SRAM: 2 KB, EEPROM: 1 KB, Clock speed: 16 MHz, Length: 45 mm, Width: 18 mm, Mass: 7 g, USB: Mini-USB Type-B, ICSP Header: Yes, DC Power Jack: No

Detailed pin configurations of the Arduino Nano microcontroller collected from official Arduino.cc website are given in Figure A2.1-3.

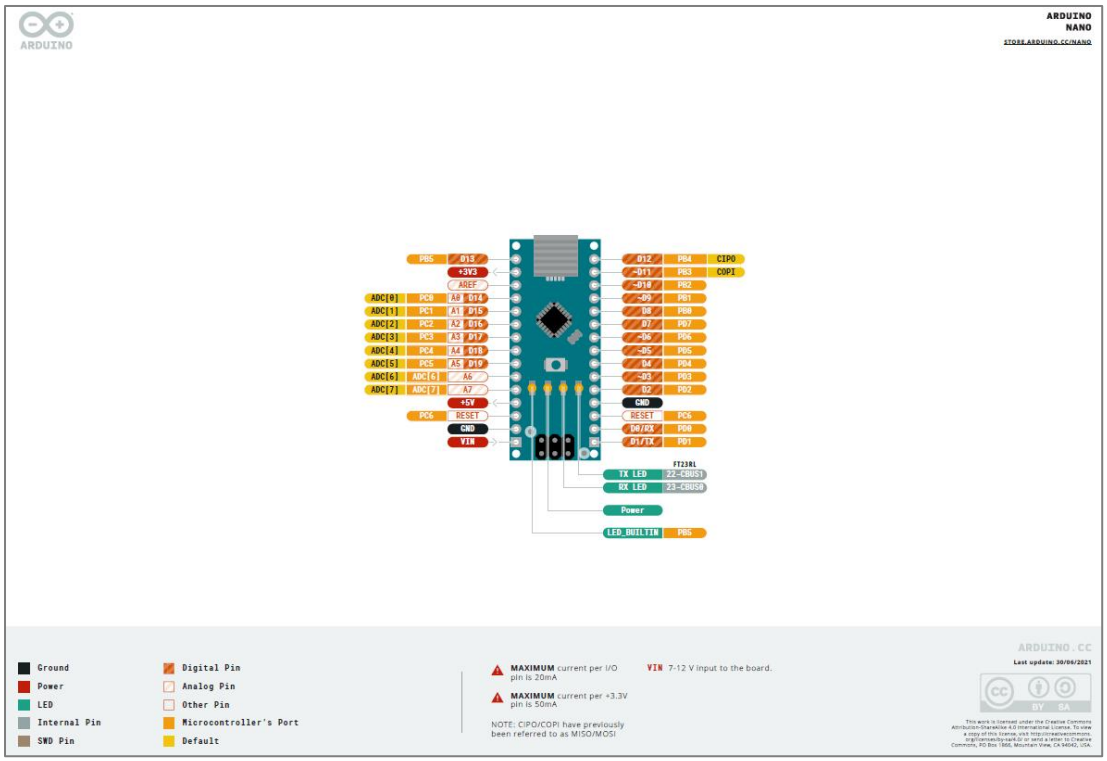

Figure A2.1: Pin diagram of Arduino Nano microcontroller Part I

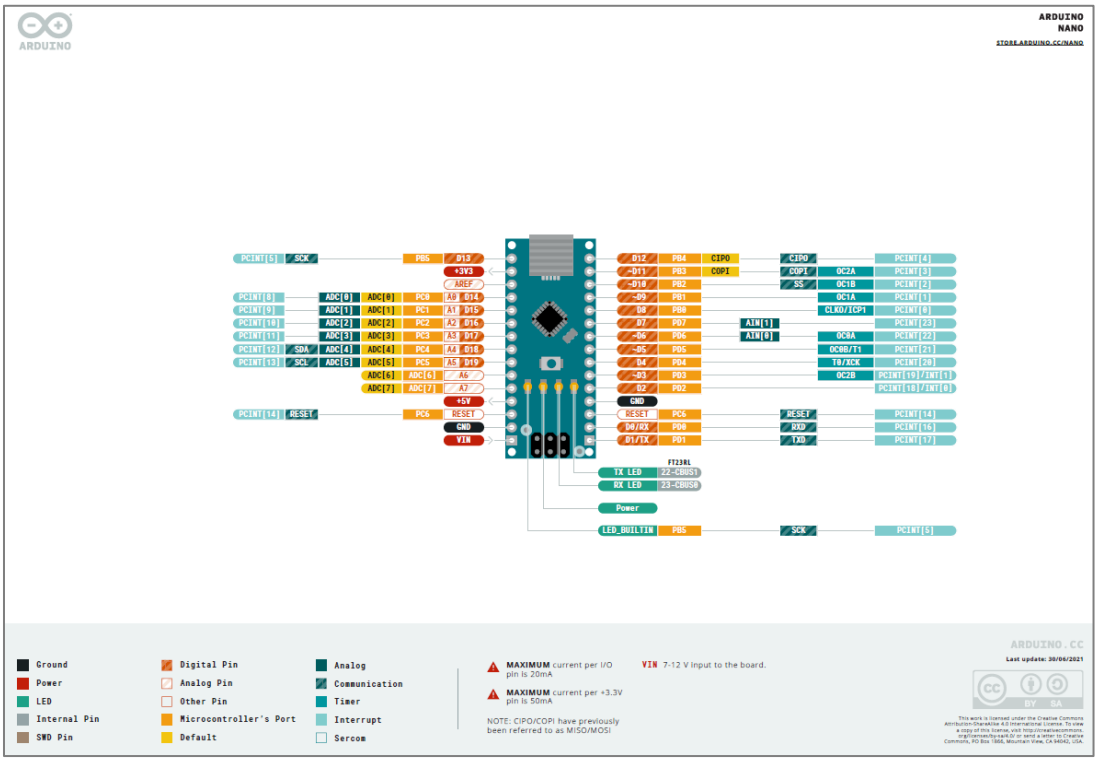

Figure A2.2: Pin diagram of Arduino Nano microcontroller Part II

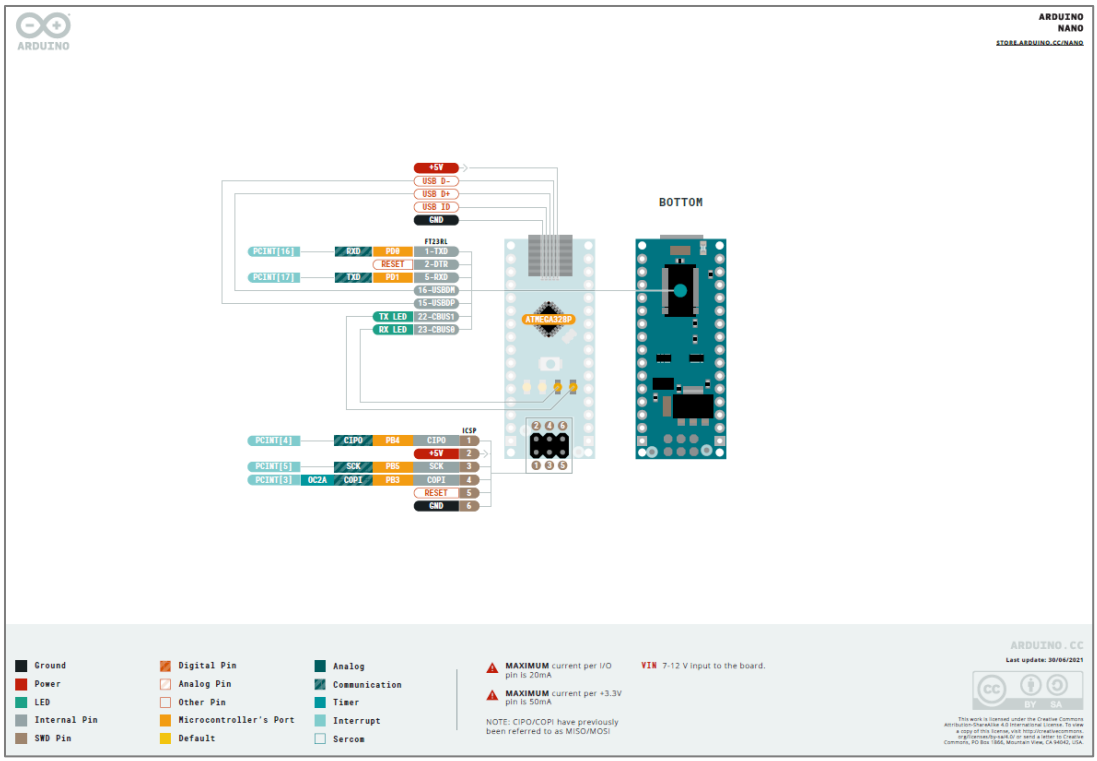

Figure A2.3: Pin diagram of Arduino Nano microcontroller Part III

AI.3 Arduino Mega 2560 Rev 3 microcontroller board

Arduino Mega 2560 Rev 3 is an open-source microcontroller board based on Microchip ATmega2560 microcontroller. The primary features of the Arduino Mega 2560 Rev 3 are: Operating Voltage : 5V, Input Voltage (recommended) : 7-12V, Input Voltage (limit): 6-20V, Digital I/O Pins: 54, PWM output pins: 15, Analog Input Pins: 16, DC Current per I/O Pin: 20 mA, DC Current for 3.3V Pin: 50 mA, Flash Memory: 256 KB, SRAM: 8 KB, EEPROM: 4 KB, Clock Speed: 16 MHz, LED_BUILTIN: 13, Length: 101.52 mm, Width: 53.3 mm, Weight: 37 g.

The detailed pin configurations of Arduino Mega 2560 Rev 3 collected from the official Arduino.cc website are given in Figure A3.1-4.

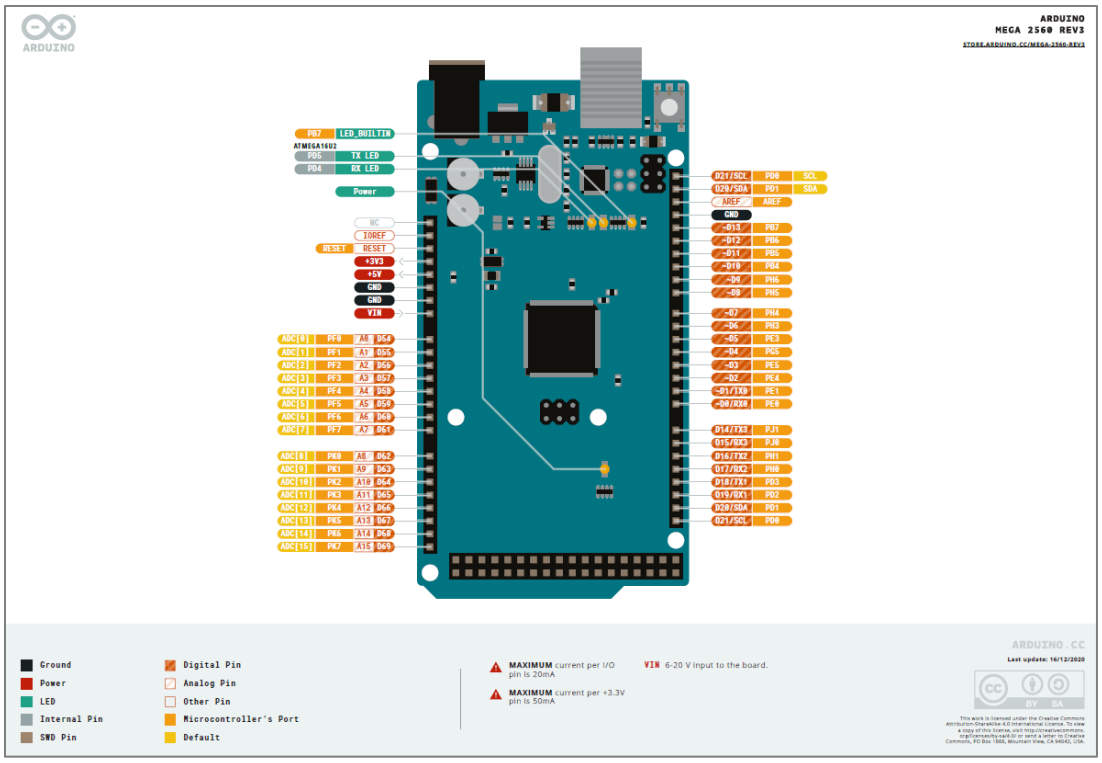

Figure A3.1: Pin diagram of Arduino Mega 2560 Rev 3 microcontroller Part I

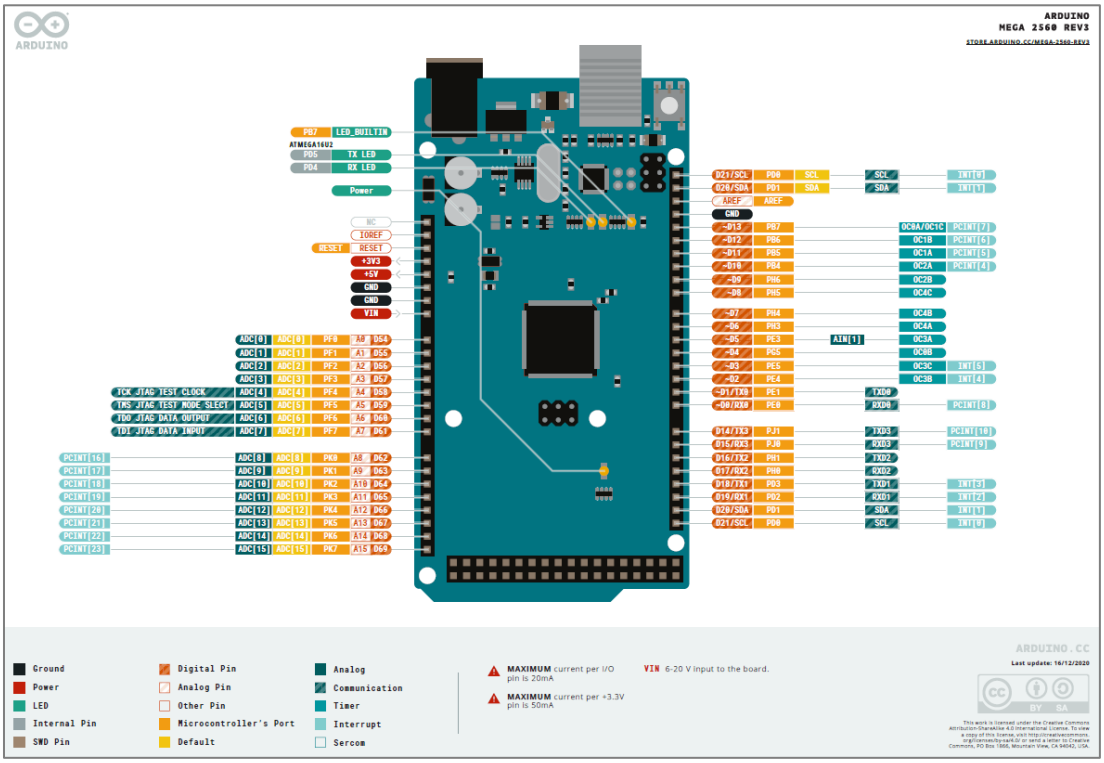

Figure A3.2: Pin diagram of Arduino Mega 2560 Rev 3 microcontroller Part II

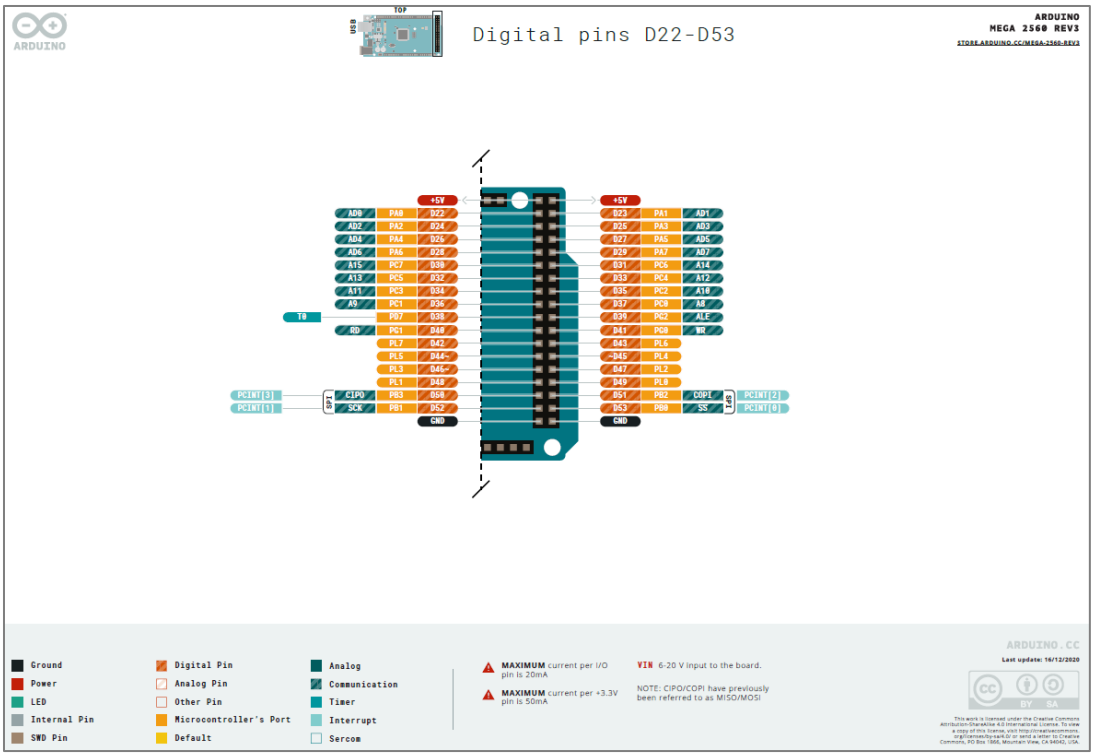

Figure A3.3: Pin diagram of Arduino Mega 2560 Rev 3 microcontroller Part III

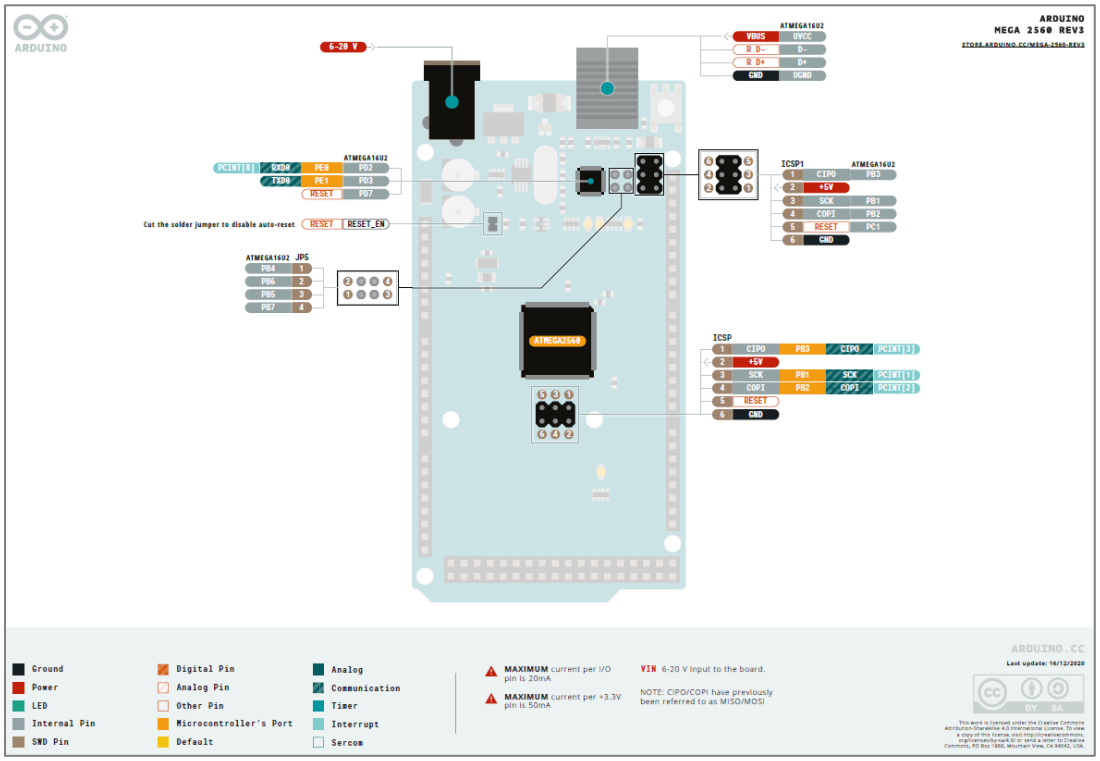

Figure A3.4: Pin diagram of Arduino Mega 2560 Rev 3 microcontroller Part IV

ANNEXURE II

Arduino based control circuit and the program used for Internally Illuminated Airlift Photobioreactor (IIAP)

AII.1. Control circuitry of the IIAP system developed using Arduino Uno Rev 3 microcontroller board.

The control circuit of the IIAP developed using Arduino Uno along with other components like 16×2 LCD display, RTC module, MoSFET switch, Relay module, Air pump, Temperature sensor, SD card reader and a DC 5 volt regulated power supply module, are shown in Figure A2.1.

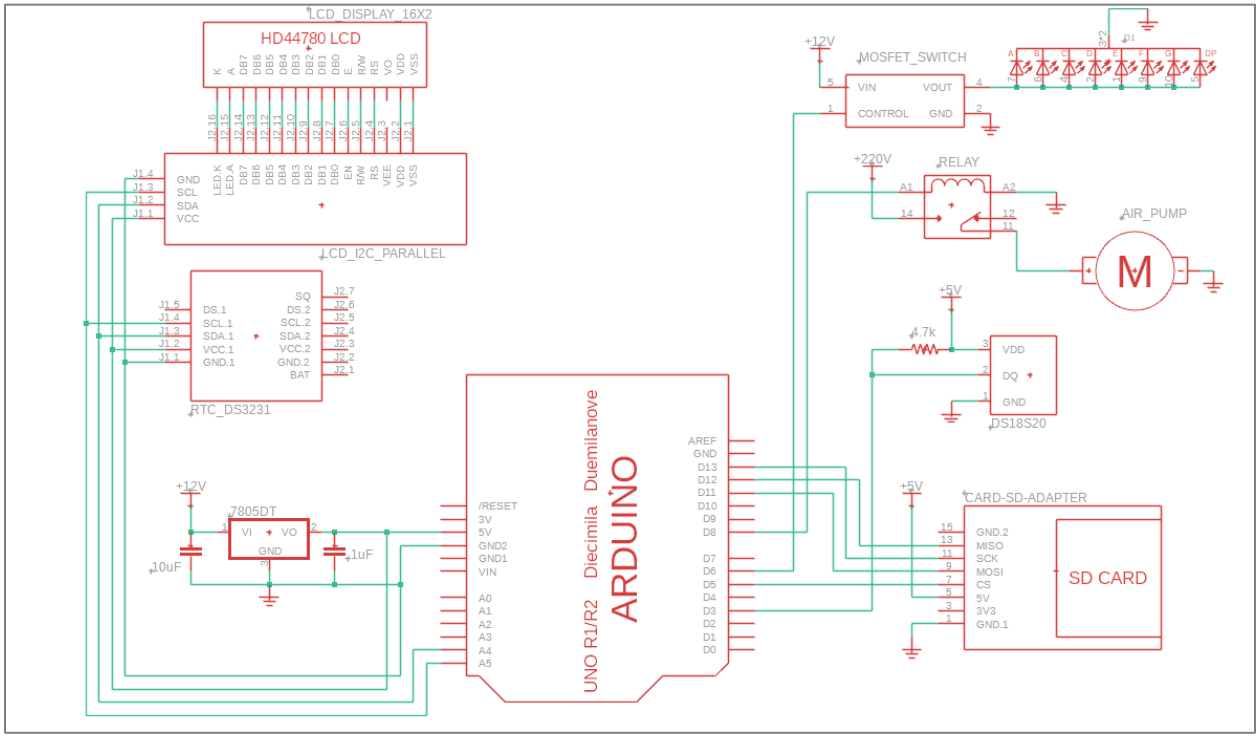

Figure A2.1: Arduino Uno based control circuitry developed for the IIAP system.

AII.2. The C++ code snippet used to control the IIAP

The Arduino Uno microcontroller was programmed using C++ code with the help of Arduino IDE, to operate the IIAP. The C++ code used is given below.

```
#include <OneWire.h>                    //Temperature Sensor header file
#include <DallasTemperature.h>
#include <SD.h>                         //SD card header file
#include <SPI.h>
#include <LiquidCrystal_I2C.h>          //LCD display header file
#include <config.h>
#include <ds3231.h>                      //RTC header file
#include <Wire.h>
```

```
#define SDcs 5                          //SD card Chip Select
#define LightPin 6                      //LED output pin PWM
#define AirPump 8                       //Air pump relay Pin
#define CO2 2                           //Carbondioxide relay Pin
#define TurbPin A2                      //Turbidity sensor Pin
#define ONE_WIRE_BUS 7                  //Temperature sensor Pin

OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
float Celcius = 0;

int logDone=1;
LiquidCrystal_I2C lcd(0x27,16,2);

struct ts t;                            //Time structure

int CultureTemp=32;
int LightDuration=18;
int LightValue;
int startYear=2021;                     //Set experiment starting time
int startMon=12;
int startDay=22;
int startHour=6;
int startMin=0;
int startSec=0;

int logInterval = 10;
char fileNameData[]="BatchOne.txt";
File myFile;

void setup() {
        lcd.init();
        lcd.backlight();
        Wire.begin();
        DS3231_init(DS3231_CONTROL_INTCN);

        sensors.begin();
        SD.begin(SDcs);

        pinMode(LightPin, OUTPUT);
        pinMode(SDcs, OUTPUT);
        pinMode(AirPump, OUTPUT);
}

void loop() {
```

```
        DS3231_get(&t);
        if((t.min%logInterval==0 && t.min!=logDone)
                {
                        logData();
                        logDone=t.min;
                }
        else    {
                runSystem();
                }
}
float getCurrent(){ }
float getpH(){ }
float getTurbidity(){ }

void logData(){
            myFile=SD.open(fileNameData,FILE_WRITE);
            myFile.print(t.year);
            myFile.print(",");
            myFile.print(t.mon);
            myFile.print(",");
            myFile.print(t.mday);
            myFile.print(",");
            myFile.print(t.hour);
            myFile.print(",");
            myFile.print(t.min);
            myFile.print(",");
            myFile.print(t.sec);
            myFile.print(",");
            myFile.print(Celcius);
            myFile.print(",");
            myFile.println(LightValue);
            myFile.close();
}
void runSystem(){
            DS3231_get(&t);
            if(t.min>=55){
                digitalWrite(AirPump,LOW);
                }
  else{
        digitalWrite(AirPump,HIGH);
   }

  sensors.requestTemperatures();                    //CONTROL TEMPERATURE
  Celcius=sensors.getTempCByIndex(0);
   if(t.hour>=6)
   {
```

```
      if(Celcius>CultureTemp){
                LightValue=150;
                analogWrite(LightPin,LightValue);
                delay(1000);
      }else{
                LightValue=250;
                analogWrite(LightPin,LightValue);
      }
   }else{
              analogWrite(LightPin,0);
   }

   lcd.setCursor(0,0);
   lcd.print(t.mday);
   lcd.print("/");
   lcd.print(t.mon);
   lcd.print("/");
   lcd.print(t.year);
   lcd.print ("  ");
   lcd.print(t.hour);
   lcd.print(":");
   lcd.print(t.min);
   lcd.setCursor(0,1);
   lcd.print("Temp:");
   lcd.print(Celcius);
   //delay(200);
   //lcd.clear();
}
```

ANNEXURE III

Arduino Mega 2560 Rev 3 based control circuit and the program used for Stacked Tray Automated Modular Photobioreactor (STAMP).

AIII.1. Control circuitry of the STAMP system developed using Arduino Mega Rev 3 microcontroller board.

The control system of the STAMP system was developed using Arduino Mega Rev 3 microcontroller board along with other electronic components like TFT display, RTC module, SD card module, relay modules to control the air and water pump, MoSFET switch to control the LED lights, temperature sensor, pH sensor, 220VAC-5V DC power source and a buzzer.

The detailed circuit diagram of the control system developed for the STAMP system is shown in Figure A3.1.

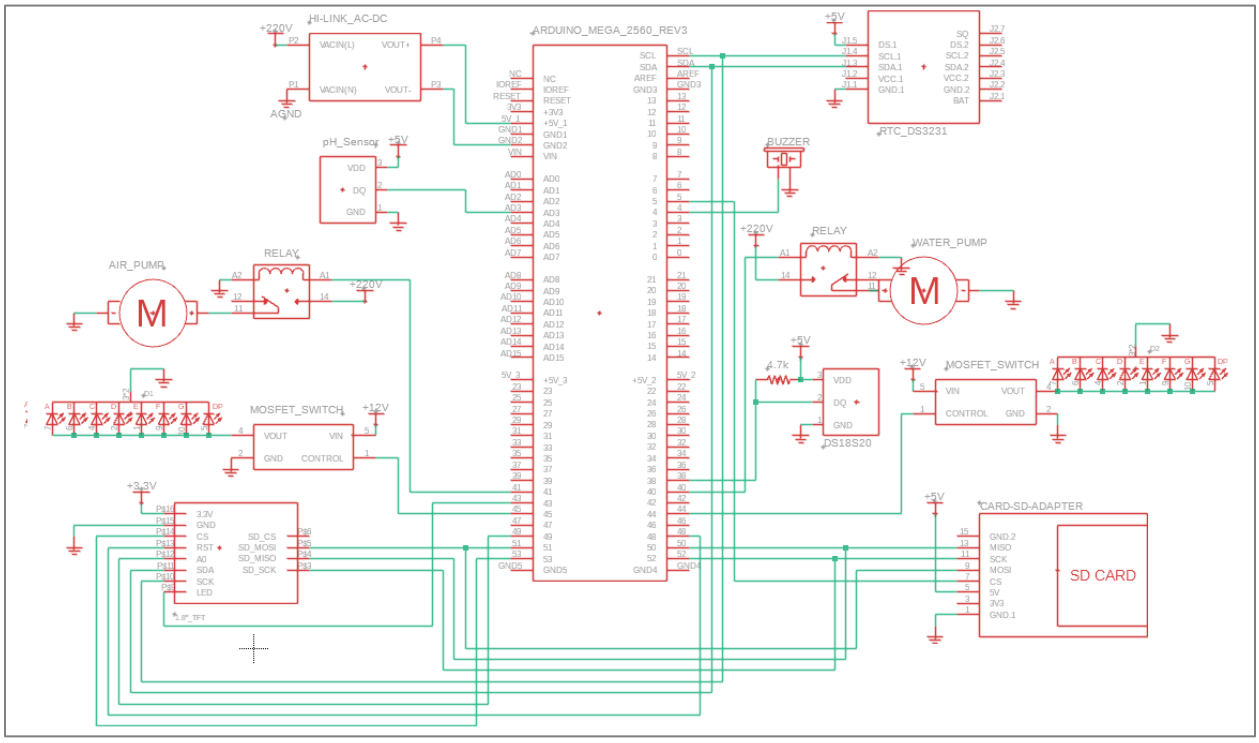

Figure A3.1: Arduino Mega 2560 Rev 3 based control system developed for the STAMP.

AIII.2. The C++ code uploaded to the Arduino Mega 2560 Rev 3 microcontroller unit to control the operations of the STAMP.

The Arduino Mega 2560 Rev 3 microcontroller was programmed using C++ code with the help of Arduino IDE, to operate the STAMP. The C++ code used is given below.

```
#include <Sodaq_wdt.h>                  //Watch Dog Timer
#include <BH1750.h>                     // Light Intensity Sensor Library
#include <OneWire.h>                     // Temperature Sensor Library
#include <DallasTemperature.h>
#include "SPI.h"
#include "Adafruit_GFX.h"
#include "Adafruit_ILI9341.h"
#include <XPT2046_Touchscreen.h>
#include <config.h>
#include <ds3231.h>
#include <Wire.h>
#include <SD.h>                         // SD card Library
#include <EEPROM.h>                     // EEPROM Library

#define ONE_WIRE_BUS 38                  // Initialize temperature sensor
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
float Celcius = 0;

struct ts t;

BH1750 lightSensor;                     // Initialize light intensity sensor

/***********************DEFIGN TFT SCREEN PINS*************************/
#define TFT_DC 49
#define TFT_CS 53
#define TFT_RST 48
#define TFT_MOSI 51
#define TFT_MISO 50
#define TFT_CLK 52
#define TFTLight 43
Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC, TFT_RST);

#define ROTATION 3

#define SensorPin A3                    //pH meter Analog output to Arduino Analog Input 0
float calibration_value = -0.5;
```

```
#define TurbidityPin A2                    //Turbidity sensor Pin

#define Buzzer 4                           //Buzzer Pin
#define SDcs 5                             //SD card Chip Select
File myFile;

#define GLightPin 44                       //LED ligth output PWM pins Growth Light
#define SLightPin 45                       //LED ligth output PWM pins Stress Light

#define WaterPump 40                       // Media circulation Pump
#define CO2Pin 41

//**************SET EXPERIMENTAL PARAMETERS*****************************

char fileNameData[]="PINKL2.txt";          //SD card file to save data

float setpH=7.5;                           //ph
int dataLogIntv=10;                        //minutes
int lightON=21;                            //hours
int waterON=21;                            //hours

int startYear=2023;                        //Set experiment Starting time
int startMon=4;
int startDay=13;
int startHour=6;
int startMin=0;
int startSec=0;

int endYear=2023;                  //Set experiment Ending time (100 hours from start time)
int endMon=5;
int endDay=14;
int endHour=9;
int endMin=0;
int endSec=0;


float disp_Celcius=0, disp_pH=0, disp_Turbidity=0;         //variables for TFT display
                                                            uint16_t disp_lightIntensity=0;


void setup() {
        pinMode(TFT_CS, OUTPUT);
        digitalWrite(TFT_CS,HIGH);
        pinMode(TFTLight,OUTPUT);
        digitalWrite(TFTLight,HIGH);
        pinMode(CO2Pin,OUTPUT);
```

```
        pinMode(WaterPump,OUTPUT);

        tft.begin();
        tft.setRotation(ROTATION);

        tft.fillScreen(ILI9341_WHITE);

        Wire.begin();
        DS3231_init(DS3231_CONTROL_INTCN);

        pinMode(Buzzer,OUTPUT);
        digitalWrite(Buzzer, HIGH);

        lightSensor.begin();                    //initialize Light sensor BH1750
        SD.begin(SDcs);                         //initialize SD card
}


void loop(){
        displayTime();
        int ExpStatus = EEPROM.read(0);
        DS3231_get(&t);
        if(ExpStatus==11){
                runExperiment();
                }
        elseif((t.year==startYear)&&(t.mon==startMon)&&(t.mday==startDay)&&(t.hour==star
        tHour)&&(t.min==startMin)){
                 EEPROM.write(0,11);
                }
        else{
                 displayMessage("0x04FF", 40,100,3,"NO EXPERIMENT");
            }
        sodaq_wdt_reset();
}

void runExperiment(){
   int logDone=1;
   bool terminate=false;
 // uint16_t CO2count=0;

   while(terminate==false){
     sodaq_wdt_reset();                         /* RESET WATCH DOG TIMER */
     DS3231_get(&t);                            //Read current time
     if(t.min%dataLogIntv==0 && t.min!=logDone){ //Log Data
        if(!SD.begin(SDcs)){
          digitalWrite(Buzzer,LOW);
```

```
  tft.fillScreen(ILI9341_WHITE);
  tft.setCursor(10,50);
  tft.setTextColor(ILI9341_RED);
  tft.setTextSize(5);
  tft.print("SD CARD ERROR");
  while(!SD.begin(SDcs)){}
  digitalWrite(Buzzer,HIGH);
 }else{
  sodaq_wdt_reset();                          /* RESET WATCH DOG TIMER */

  float Celcius=Measure_Temp();
  float pH=Measure_pH();
  float Turbidity=Measure_Turbidity();
  uint16_t lightIntensity=lightSensor.readLightLevel(true);
  sodaq_wdt_reset();                          /* RESET WATCH DOG TIMER */

  myFile=SD.open(fileNameData,FILE_WRITE);
  myFile.print(t.year);
  myFile.print(",");
  myFile.print(t.mon);
  myFile.print(",");
  myFile.print(t.mday);
  myFile.print(",");
  myFile.print(t.hour);
  myFile.print(",");
  myFile.print(t.min);
  myFile.print(",");
  myFile.print(t.sec);
  myFile.print(",");
  myFile.print(Celcius);
  myFile.print(",");
  myFile.print(pH);
  myFile.print(",");
  myFile.print(Turbidity);
  myFile.print(",");
  myFile.print(lightIntensity);
// myFile.print(",");
// myFile.print(CO2count);
  myFile.println(";");
  myFile.close();

//   CO2count=0;
 }
logDone=t.min;
sodaq_wdt_reset();                            /* RESET WATCH DOG TIMER */
tft.fillScreen(ILI9341_WHITE);
```

```
    }
    else{                                   //CONTINUE RUNNING EXPERIMENT
        DS3231_get(&t);

if((t.year==endYear)&&(t.mon==endMon)&&(t.mday==endDay)&&(t.hour==endHour)&&(t.m
in==endMin)){
            EEPROM.write(0,00);
            terminate=true;
            digitalWrite(WaterPump,LOW);
            turnOFFLED();
            digitalWrite(CO2Pin,LOW);
            digitalWrite(Buzzer,LOW);
            delay(3000);
            digitalWrite(Buzzer,HIGH);

          }
          else{
              int lightState=0;                 //TURN ON LIGHTS
              for(int i=0;i<lightON;i++){
                 int buffH=startHour+i;
                 if(buffH>=24){
                   buffH-=24;
                 }
                 if(t.hour==buffH){
                   lightState=10;
                   break;
                 }
                 else{
                   lightState=0;
                 }
               }
              if(lightState>0){
                turnONLED();
              }else{turnOFFLED();}


              bool pumpState=0;                 //TURN ON Water PUMP
              for(int i=0;i<waterON;i++){
                int buffH=startHour+i;
                if(buffH>=24){
                  buffH-=24;
                }
                if(t.hour==buffH){
                  pumpState=HIGH;
                  break;
                }else{
```

```
                pumpState=LOW;
              }
            }
            digitalWrite(WaterPump,pumpState);


 /*          float pH=Measure_pH();                    //CONTROL pH
             if(pH > setpH){
              digitalWrite(CO2Pin,HIGH);
              delay(8000);
              CO2count+=1;
             }else{
              digitalWrite(CO2Pin,LOW);
              delay(500);
             }
*/
////////////////////////////////////DISPLAY DATA IN THE TFT SCREEN////////////////////////////////////
            tft.fillScreen(ILI9341_WHITE);
            displayTime();
            tft.setTextSize(3);

            tft.setCursor(10,40);
            tft.setTextColor(ILI9341_BLUE);
            tft.print("Temp: ");
            tft.setTextColor(ILI9341_WHITE);
            tft.setCursor(100,40);
            tft.print(disp_Celcius);

            tft.setCursor(10,80);
            tft.setTextColor(ILI9341_PINK);
            tft.print("Turb: ");
            tft.setTextColor(ILI9341_WHITE);
            tft.setCursor(100,80);
            tft.print(disp_Turbidity);

            tft.setCursor(10,120);
            tft.setTextColor(0x04FF);
            tft.print("pH: ");
            tft.setTextColor(ILI9341_WHITE);
            tft.setCursor(100,120);
            tft.print(disp_pH);

            tft.setCursor(10,160);
            tft.setTextColor(ILI9341_PINK);
            tft.print("Lux: ");
            tft.setTextColor(ILI9341_WHITE);
```

```
tft.setCursor(100,160);
tft.print(disp_lightIntensity);

tft.setTextSize(2);
tft.setCursor(10,190);
tft.setTextColor(ILI9341_WHITE);
tft.print("End at: ");
tft.print(endDay);
tft.print(":");
tft.print(endMon);
tft.print(" ");
tft.print(endHour);
tft.print(":");
tft.print(endMin);


disp_Celcius=Measure_Temp();
disp_pH=Measure_pH();
disp_Turbidity=Measure_Turbidity();
disp_lightIntensity=lightSensor.readLightLevel(true);

tft.setTextSize(3);

tft.setCursor(10,40);
tft.setTextColor(ILI9341_BLUE);
tft.print("Temp: ");
tft.setCursor(100,40);
tft.print(disp_Celcius);

tft.setCursor(10,80);
tft.setTextColor(ILI9341_PINK);
tft.print("Turb: ");
tft.setCursor(100,80);
tft.print(disp_Turbidity);

tft.setCursor(10,120);
tft.setTextColor(0x04FF);
tft.print("pH: ");
tft.setCursor(100,120);
tft.print(disp_pH);

tft.setCursor(10,160);
tft.setTextColor(ILI9341_PINK);
tft.print("Lux: ");
tft.setCursor(100,160);
tft.print(disp_lightIntensity);
```

```
                tft.setTextSize(2);
                tft.setCursor(10,220);
                tft.setTextColor(ILI9341_RED);
                tft.print("End at: ");
                tft.print(endDay);
                tft.print(":");
                tft.print(endMon);
                tft.print(" ");
                tft.print(endHour);
                tft.print(":");
                tft.print(endMin);
                delay(1000);
              }
      }

    }

}


/**********************FUNCTION DEFINATIONS**************************/

float Measure_Temp(){
          sensors.requestTemperatures();
          float Celcius=sensors.getTempCByIndex(0);
          return(Celcius);
}

float Measure_pH(){
  int buf[10];
  for(int i=0;i<10;i++)       //Get 10 sample value from the sensor for smooth the value
  {
    buf[i]=analogRead(SensorPin);
    delay(10);
  }
  for(int i=0;i<9;i++)        //sort the analog from small to large
  {
    for(int j=i+1;j<10;j++)
    {
      if(buf[i]>buf[j])
      {
        int temp=buf[i];
        buf[i]=buf[j];
        buf[j]=temp;
      }
```

```
    }
  }
  unsigned long int avgValue=0;
  for(int i=2;i<8;i++){                          //take the average value of 6 center sample
   avgValue+=buf[i];
  }
  float phValue=(float)avgValue*5.0/1024/6;          //convert the analog into millivolt
  phValue=(3.5*phValue)+ calibration_value;          //convert the millivolt into pH value
  return(phValue);
}

float Measure_Turbidity(){
    int buf[10],temp;
    for(int i=0;i<10;i++)                  //Get 10 sample value from the sensor for smooth the value
      {
       buf[i]=analogRead(TurbidityPin);
       delay(10);
      }
      for(int i=0;i<9;i++)                 //sort the analog from small to large
      {
       for(int j=i+1;j<10;j++)
       {
        if(buf[i]>buf[j])
        {
          temp=buf[i];
          buf[i]=buf[j];
          buf[j]=temp;
        }
       }
      }
      unsigned long int avgValue=0;
      for(int i=2;i<8;i++)                           //take the average value of 6 center sample
        avgValue+=buf[i];
      float turbidity=(float)avgValue*5.0/1024/6;          //convert the analog into millivolt
        turbidity*=33;
        turbidity-=24;
     return(turbidity);
}


void displayTime(){
    DS3231_get(&t);
    tft.setCursor(40,10);
    tft.setTextColor(ILI9341_PINK);
    tft.setTextSize(2);
    tft.print(t.mday);
```

```
    tft.print("/");
    tft.print(t.mon);
    tft.print("/");
    tft.print(t.year);
    tft.setCursor(190,10);
    tft.print(t.hour);
    tft.print(":");
    tft.print(t.min);
    tft.print(":");
    tft.print(t.sec);
    delay(500);
/*  tft.setCursor(40,10);
    tft.setTextColor(ILI9341_WHITE);
    tft.setTextSize(2);
    tft.print(t.mday);
    tft.print("/");
    tft.print(t.mon);
    tft.print("/");
    tft.print(t.year);
    tft.setCursor(190,40);
    tft.print(t.hour);
    tft.print(":");
    tft.print(t.min);
    tft.print(":");
    tft.print(t.sec);
*/
}

void turnONLED(){                   //Set Light Intensity 0-255 PWM
  analogWrite(GLightPin,200);
  analogWrite(SLightPin,0);
}
int turnOFFLED(){                   //Turn of LED illumination
  analogWrite(GLightPin,0);
  analogWrite(SLightPin,0);
}

void displayMessage(char color[20], int x, int y, int siz, char message[30]){
    tft.fillScreen(ILI9341_WHITE);
    tft.setCursor(x,y);
    tft.setTextColor(color);
    tft.setTextSize(siz);
    tft.print(message);
    delay(1000);
}
```

ANNEXURE IV

Arduino Nano microcontroller based control unit and the C++ code for the microalgae culture experimental setup.

AIV.1. Control circuitry of the microalgae culture experimental setup developed using Arduino Nano microcontroller board.

The control system of the microalgae culture experimental setup was developed using Arduino Nano microcontroller board along with other electronic components like 20×4 LCD display, 4×4 keypad, RTC module, SD card module, relay modules to control the air pump, MoSFET switch to control the LED lights, temperature sensor, color sensor, light intensity sensor, cooling module, 5V DC power source and a buzzer.

The detailed circuit diagram of the developed control system is shown in Figure A4.1.

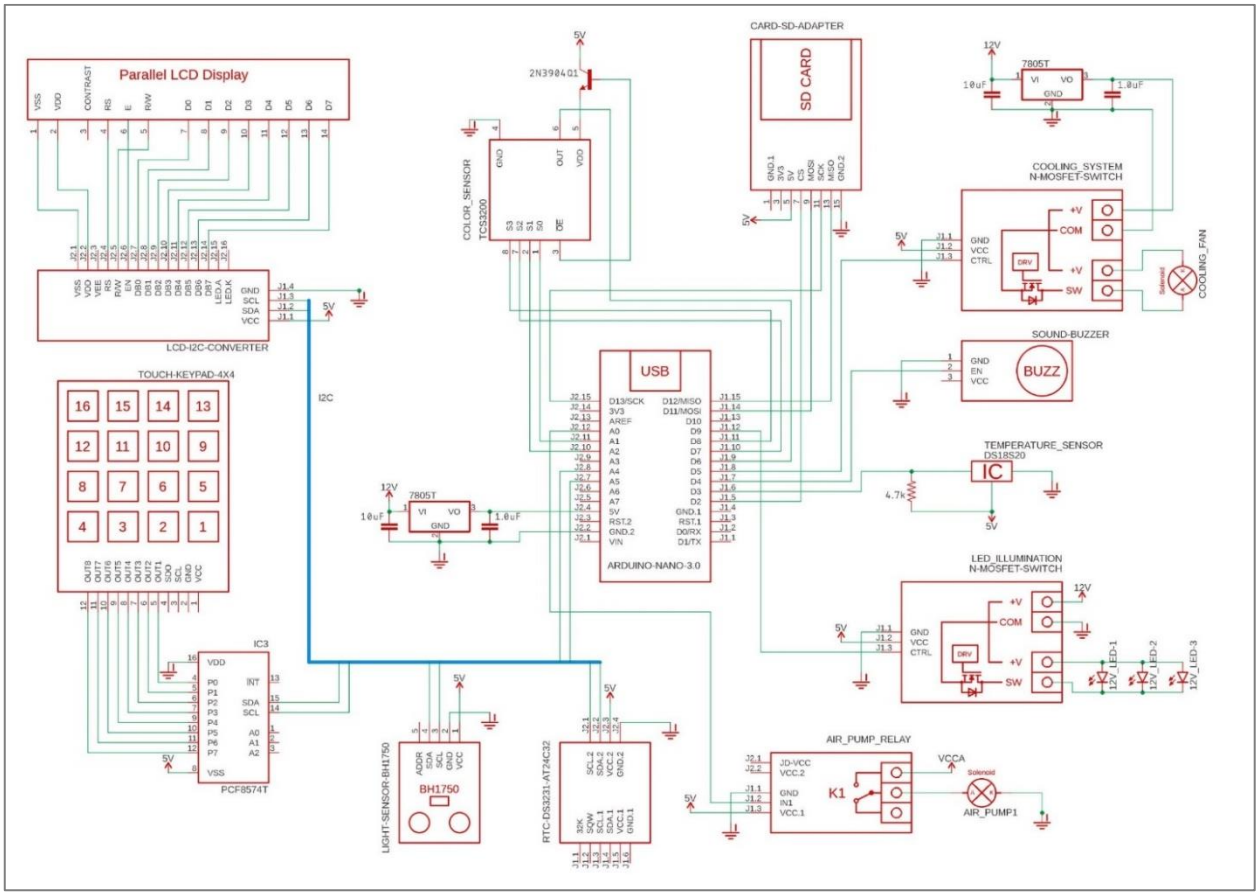

Figure A4.1: Arduino Nano based control system developed for the microalgae culture experimental setup.

AIV.2. The C++ code uploaded to the Arduino Nano microcontroller unit to control the operations of the microalgae culture experiment test setup.

The Arduino Nano microcontroller was programmed using C++ code with the help of Arduino IDE, to operate the microalgae culture experimental setup. The C++ code used is given below.

```
#include <Sodaq_wdt.h>
#include <OneWire.h>                          // Temperature Sensor Library
#include <DallasTemperature.h>
#include <BH1750.h>                           // Light Intensity Sensor Library
#include <SD.h>                               // SD card Library
#include <SPI.h>                              // SPI library to interface SD card
#include <EEPROM.h>                           // EEPROM Library
#include <Wire.h>                             // Wire Library for serial communication
#include <config.h>                           // RTC Library
#include <ds3231.h>
#include "I2CKeyPad.h"                        // I2C keypad Library
#include <LiquidCrystal_I2C.h>                 // I2C LCD Library

#define SDcs 2                                //SD card Chip Select
#define LightPin 9                            // Pin to control the LEDs
#define AirPump A0                            // Pin to control the Air pumps
#define Cooler 5                              // Pin to control the Cooling system
#define Buzzer 4                              // Pin to controll the buzzer Alarm

LiquidCrystal_I2C lcd(0x27,20,4);             //initialize 20*4 LCD

const uint8_t KEYPAD_ADDRESS = 0x20;          //initialuze keypad
I2CKeyPad keyPad;

struct ts t;                                  // RTC time structure

#define ONE_WIRE_BUS 3                        // Initialize temperature sensor
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
float Celcius = 0;

BH1750 lightSensor;                           // Initialize light intensity sensor

File myFile;

//**************SET EXPERIMENTAL PARAMETERS*****************************

char fileNameData[]="RedG.txt";               //SD card file to save data

int lightIntensity(){                         //Set Light Intensity 0-255 PWM
```

```
  return(230);
}

float tempR=30;                                    //celcius
int dataLogIntv=10;                                //minutes
int lightON=15;                                    //hours
int airON=15;                                      //hours

int startYear=2023;                                //Set experiment Starting time
int startMon=2;
int startDay=17;
int startHour=17;
int startMin=5;
int startSec=0;

int endYear=2023;                  //Set experiment Ending time (100 hours from start time)
int endMon=3;
int endDay=18;
int endHour=9;
int endMin=0;
int endSec=0;

bool lcdBacklight = true;

void setup() {
   Wire.begin();
   Serial.begin(9600);

   keyPad.begin(KEYPAD_ADDRESS);

   lcd.init();
   lcd.backlight();

   DS3231_init(DS3231_CONTROL_INTCN);              //initialize RTC

   sensors.begin();                                //initialize Temperature sensor

   pinMode(SDcs,OUTPUT);
   pinMode(LightPin,OUTPUT);
   pinMode(10,OUTPUT);
   pinMode(AirPump,OUTPUT);
   pinMode(Cooler,OUTPUT);
   pinMode(Buzzer,OUTPUT);

   lightSensor.begin();                    //initialize Light sensor BH1750
```

```
    SD.begin(SDcs);

    analogWrite(Cooler,0);

    sodaq_wdt_enable(WDT_PERIOD_8X);
}

void loop() {
      displayDateTime(0,0);
      int ExpStatus = EEPROM.read(0);
      DS3231_get(&t);
      if(ExpStatus==11){
        runExperiment();
      }else
if((t.year==startYear)&&(t.mon==startMon)&&(t.mday==startDay)&&(t.hour==startHour)&&(
t.min==startMin)){
        EEPROM.write(0,11);
      }else{
        lcd.setCursor(2,2);
        lcd.print("No Experiment");
      }
      sodaq_wdt_reset();                          /* RESET WATCH DOG TIMER */
}

void runExperiment(){
   if(!SD.begin(SDcs)){
      lcd.clear();
      displayMessage(0,1,"SD CARD ERROR");
      digitalWrite(Buzzer,HIGH);
      while(!SD.begin(SDcs)){}
      digitalWrite(Buzzer,LOW);
      lcd.clear();
    }
  int LightValue=lightIntensity();
  int logDone=1;
  bool terminate=false;
  long lcdTime=0;
  bool lcdStatus=true;
  while(terminate==false){
      sodaq_wdt_reset();                          /* RESET WATCH DOG TIMER */
      DS3231_get(&t);                                          //Read current time
      if(t.min%dataLogIntv==0 && t.min!=logDone){              //LOG DATA
          if(SD.begin(SDcs)==false){
            lcd.clear();
            displayMessage(0,1,"SD card ERROR");
            digitalWrite(Buzzer,HIGH);
```

```
        while(SD.begin(SDcs)==false){}
        digitalWrite(Buzzer,LOW);
       }else{
        sodaq_wdt_reset();                          /* RESET WATCH DOG TIMER */
        uint16_t lightIntensity=lightSensor.readLightLevel(true);
        sensors.requestTemperatures();
        Celcius=sensors.getTempCByIndex(0);
        sodaq_wdt_reset();                          /* RESET WATCH DOG TIMER */

        myFile=SD.open(fileNameData,FILE_WRITE);
        myFile.print(t.year);
        myFile.print(",");
        myFile.print(t.mon);
        myFile.print(",");
        myFile.print(t.mday);
        myFile.print(",");
        myFile.print(t.hour);
        myFile.print(",");
        myFile.print(t.min);
        myFile.print(",");
        myFile.print(t.sec);
        myFile.print(",");
        myFile.print(lightIntensity);
        myFile.print(",");
        myFile.print(Celcius);
        myFile.print(",");
        myFile.close();
       }
      logDone=t.min;
    }
    else{                                   //CONTINUE RUNNING EXPERIMENT
        DS3231_get(&t);

if((t.year==endYear)&&(t.mon==endMon)&&(t.mday==endDay)&&(t.hour==endHour)&&(t.m
in==endMin)){
            EEPROM.write(0,00);
            terminate=true;
            digitalWrite(AirPump,LOW);
            analogWrite(LightPin,0);
            digitalWrite(Buzzer,HIGH);
            delay(3000);
            digitalWrite(Buzzer,LOW);
            lcd.clear();
          }
          else{
              int lightState=0;                 //TURN ON LIGHTS
```

```
for(int i=0;i<lightON;i++){
   int buffH=startHour+i;
   if(buffH>=24){
    buffH-=24;
   }
   if(t.hour==buffH){
    lightState=LightValue;
    break;
   }
   else{
    lightState=0;
   }
 }
analogWrite(LightPin,lightState);
int airState=0;                          //TURN ON AIR PUMP
for(int i=0;i<airON;i++){
  int buffH=startHour+i;
  if(buffH>=24){
   buffH-=24;
  }
  if(t.hour==buffH){
   airState=HIGH;
   break;
  }else{
   airState=LOW;
  }
}
digitalWrite(AirPump,airState);
sensors.requestTemperatures();                  //CONTROL TEMPERATURE
Celcius=sensors.getTempCByIndex(0);
  if(Celcius > tempR){
   analogWrite(Cooler,250);
   delay(500);
  }else{
   digitalWrite(Cooler,LOW);
   delay(500);
  }
displayDateTime(0,0);
displayMessage(1,1,"End on: ");
lcd.print(endDay);
lcd.print(":");
lcd.print(endMon);
lcd.print(" ");
lcd.print(endHour);
lcd.print(":");
lcd.print(endMin);
```

```
                lcd.setCursor(0,2);
                lcd.print("Temperature: ");
                lcd.print(Celcius);
                lcd.setCursor(18,2);
                lcd.print("'C");
                lcd.setCursor(0,3);
                lcd.print("Light: ");
                lcd.print(lightSensor.readLightLevel(true));
                lcd.print(" lux");
              }
     }

    if(keyPad.isPressed()==true){
       if(getChar()=='C'){
          if(lcdBacklight==true){
            lcdBacklight=false;
          }
          else if(lcdBacklight==false){
            lcdBacklight=true;
          }
       }
       if(lcdBacklight==true){
          lcd.backlight();
       }
       else if(lcdBacklight==false){
           lcd.noBacklight();
       }
    }
  }
  lcd.backlight();
}

char getChar(){
  sodaq_wdt_reset();                          /* RESET WATCH DOG TIMER */
//char keys[] = "123A456B789C*0#DNF";         /*<----Gum key PAD*/
  char keys[] = "147*2580369#ABCDNF";         /*<----Telephone key PAD*/
  uint8_t idx = keyPad.getKey();
  char key = keys[idx];
  return(key);
  delay(500);
}

void displayMessage(int x, int y, char z[]){
  lcd.setCursor(x,y);
  lcd.print(z);
}
```

```
void displayDateTime(int x, int y){
    DS3231_get(&t);
    lcd.setCursor(x,y);
    if(t.mday<10){
      lcd.print('0');
      lcd.print(t.mday);
    }else{
    lcd.print(t.mday);}
    lcd.print("/");
    if(t.mon<10){
      lcd.print('0');
      lcd.print(t.mon);
    }else{
    lcd.print(t.mon);}
    lcd.print("/");
    lcd.print(t.year);
    lcd.print ("  ");
    if(t.hour<10){
      lcd.print('0');
      lcd.print(t.hour);
    }else{
    lcd.print(t.hour);}
    lcd.print(":");
      if(t.min<10){
      lcd.print('0');
      lcd.print(t.min);
    }else{
    lcd.print(t.min);}
    lcd.print(":");
    if(t.sec<10){
      lcd.print('0');
      lcd.print(t.sec);
    }else{
    lcd.print(t.sec);}
    delay(1000);
}
```

LIST OF PUBLICATIONS

1. **Borah, D.**, Eldiehy, K. S., Hatiboruah, D., Mandal, M., & Deka, D. (2022). *An Integrated Approach for Simultaneous Monitoring and Data Acquisition on the Culture of Green Microalga Chlorella homosphaera Using Different LED Illumination*. BioEnergy Research, 16(1): p. 601-610.
2. **Borah, D.**, Eldiehy K. S., Deka, D., *Optoelectronic sensitivity-based investigation on LED inspired microalgae cultivation.* In the proceedings of International Symposium on Advances in Algal Research (AAR-2023), pp.45-47, held at IIT Guwahati, Assam, during 12-14 June 2023.
3. **Borah, D.**, Eldiehy K. S., Kataki, R., Deka, D., *Circular bio-based economy of microalgae-based processes and products*. Algal Bioreactors Science, Engineering and Technology of upstream processes. Vol 1, Pages 39-67, Elsevier.
4. Eldiehy, K. S., Gohain, M., Daimary, N., **Borah, D**., Mandal, M., & Deka, D. (2022). *Radish (Raphanus sativus L.) leaves: A novel source for a highly efficient heterogeneous base catalyst for biodiesel production using waste soybean cooking oil and Scenedesmus obliquus oil*. Renewable Energy, 191, 888-901.
5. Eldiehy, K. S., Daimary, N., **Borah, D**., Sarmah, D., Bora, U., Mandal, M., & Deka, D. (2022). *Towards biodiesel sustainability: Waste sweet potato leaves as a green heterogeneous catalyst for biodiesel production using microalgal oil and waste cooking oil.* Industrial Crops and Products, 187, 115467.
6. Eldiehy, K. S., Bardhan, P., **Borah, D**., Gohain, M., Rather, M. A., Deka, D., & Mandal, M. (2022). *A comprehensive review on microalgal biomass production and processing for biodiesel production.* Fuel, 324, 124773.
7. Eldiehy, K. S., Bardhan, P., **Borah, D**., Rather, M. A., Chutia, H., Bhagya Raj, G. V., Mandal, M., & Deka, D. (2022). *Optimization of nutrient composition for enhanced microalgal biomass and macromolecules using RSM: An integrated approach towards improving microalgal biodiesel feasibility.* Journal of Applied Phycology, 1-14.
8. Eldiehy, K. S., Das, V., **Borah, D**., Mandal, M., & Deka, D. (2021). *Strategic approaches to enhance lipid accumulation in microalgae for bioprospecting.* In Dalai A.K., Goud, V.V.,

List of Publications

Nanda, S., and Borugadda, V.B. editors, Algal Biorefinery Developments, Challenges and Opportunities, pages 50-84, Taylor & Francis Group, England, UK, 2021.