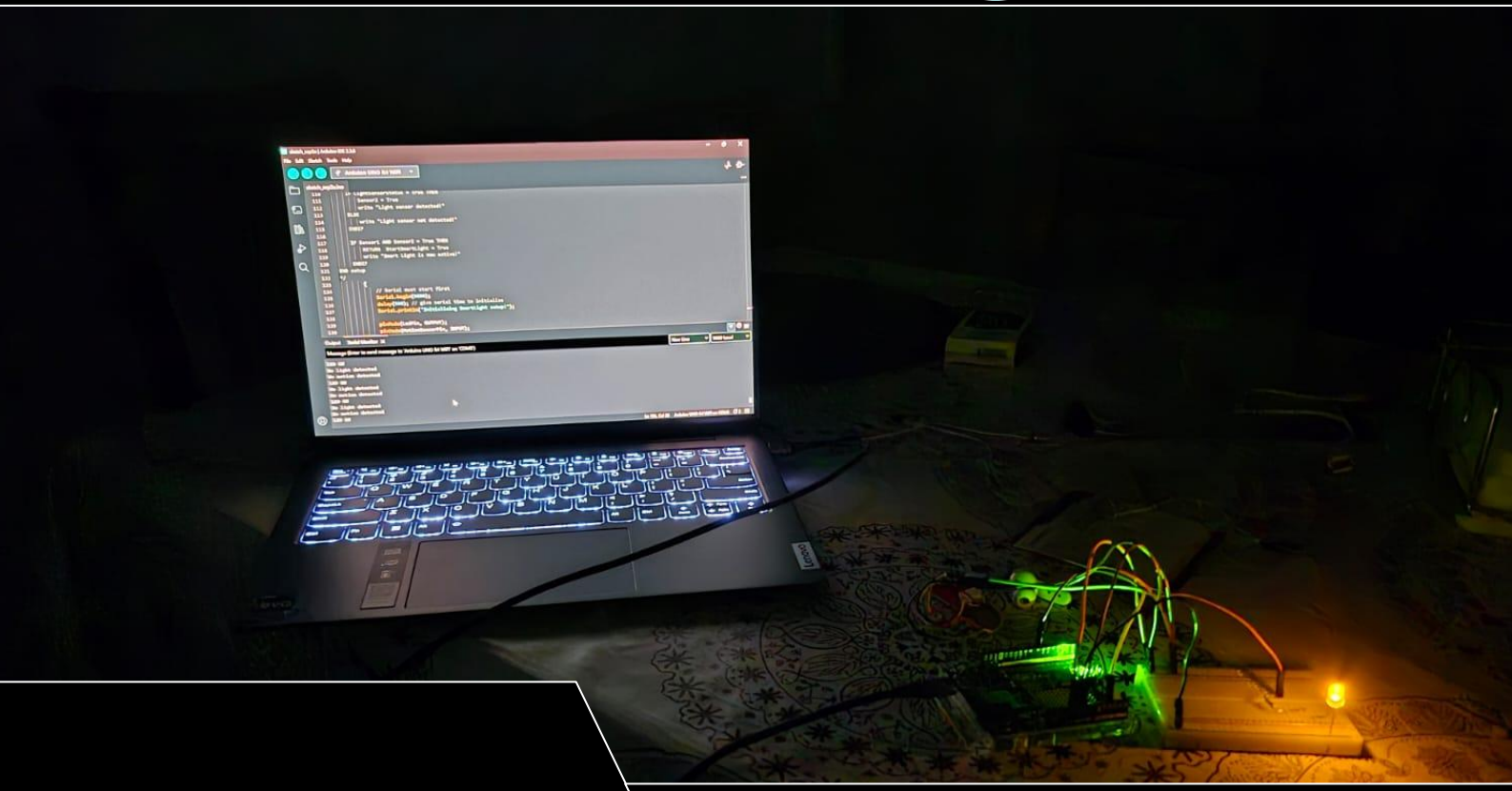
SE2026



Assessment Task 3 - Smart light

Mechatronics and OOP Folio

```
//class public variables to declare type
// booleon values which will be determined by functions which will return bool/float
bool LEDLight;
double LightLevel;
bool Light;
bool LightSensorStateOn;
bool StartSmartLight;
// pins constant soo they cannot be changed
const int LedPin;
const int PhotoResistorPin;
SmartLight() // constructor -> __init__ in python to assign values to class + pin and default bool assign
    : PhotoResistorPin(A2),
      LedPin(D7)
{
    // starter values each time class is called (for safety)
    StartSmartLight = false;
    LightSensorStateOn = true;
    LEDLight = false;
    LightLevel = 0.0;
    Light = false;
}
bool LightSensor(){LightLevel = analogRead(PhotoResistorPin); //get light level from pin reading
    if (LightLevel > 150){// let light = true if reading above 150
        Serial.println("Sensor is detecting light!");
        return Light = true;}//otherwise return false if light < 150
        Serial.println("No light detected");
        return Light = false;}
 bool setup(){
    Serial.begin(9600);
    delay(500); // give serial time to initialize
    Serial.println("Initializing SmartLight setup!");
    pinMode( pinNumLedPin,  pinModOUTPUT);
    LightSensor();
    if (LightSensorStateOn){Serial.println("Light Sensor is ON!");return StartSmartLight = true;}Serial.println("Light Sensor is OFF");return StartSmartLight = false;}
```

by Amit Singh

Table of contents


DEVELOPMENT
LOG

PROJECT LOG - 27/08/25

- Today I received information about the assessment
  - What is required from the task
  - Due date

- I plan on getting my project idea down and finish the planning phase so I can start development and research all requirements to successfully complete the assessment.

- I want to do something home related so this way the project I make will mainly serve as a quality of life product for consumers and increase automation, whilst encouraging to be a eco friendly solution.

- I decided to go with a smart light system as the scope to success seemed achievable within the given time frame.

- Completed statement of intent

- I started researching different tools i can use and components I would need to make this work or at least simulate how it would work.

- I also finished setting up my work environment using tools such as clion with the platformIO plugin to help me code in arduino for easy development.

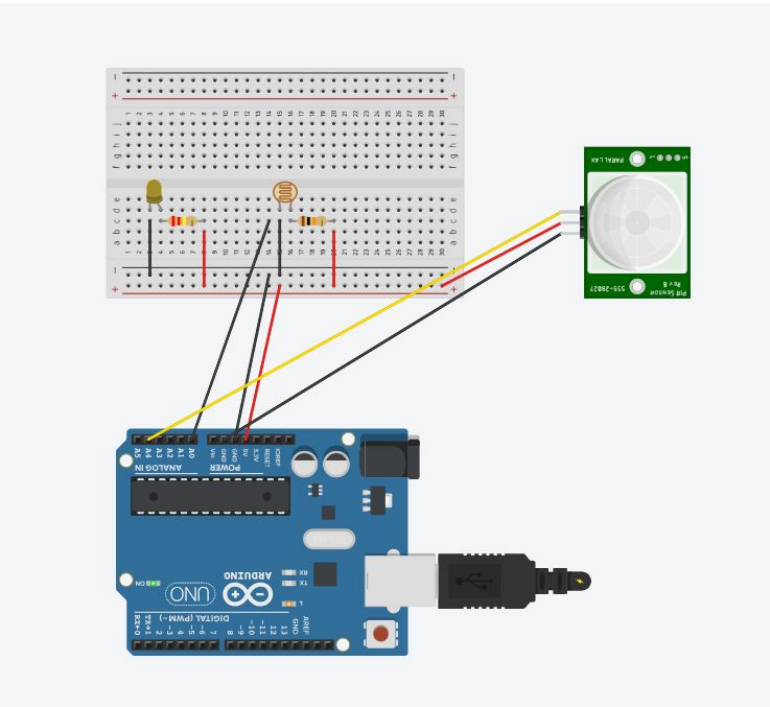
PROJECT LOG - 28/08/25

- Today I plan on researching components needed for the project.
- Also will try to improve my c++ oop skills as I am new to the language.

- Read cpp and arduino documentation
  - docs.arduino.cc
  - cppreference.com
- FInished writing layout of my report template

- Thinking if I need further initial research or if I should proceed to Development of the project

- Decided to start developing a diagram for my project. (I don’t know what I am doing)
- This looks correct I hope:



PROJECT LOG - 29/08/25

- Today I plan on obtaining the arduino components from school
    - I realised that I forgot to get male-male and male-female jumper wires oh well now I can't test my design until tuesday.

- Decided to continue researching and before I even consider touching the real board I plan to ensure that my design works the way it is intended to without any chance of breaking or damaging the real components.

- I decided continue working on development of the arduino project working on the functionality concept and final design of board.

- FInished development of arduino project section of folio however I might come back to tweak it if needed.

- Also finished steps to success

- Tomorrow I plan to start my algorithm development and desk checking process before I began coding the actual project.

- Hoping it goes well without too many errors

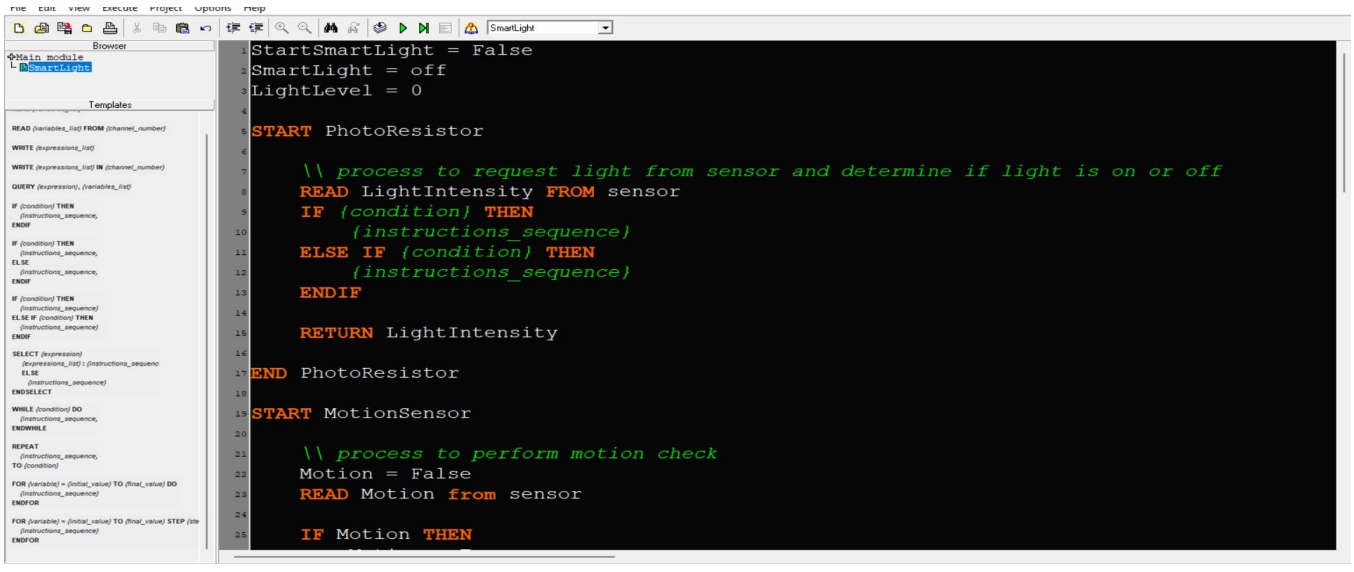
PROJECT LOG - 30/08/25

- Today I plan on starting my algorithm design for pseudocode to determine how my app with function
  - I started by planning all logic and functions that will be needed for my project to work
    - Such as sensor selection statements
    - Iteration
    - Basic functions
    - Data
    - Variable names

- Began using larp for auto index and highlight important keywords (this should increase the speed of the development process and allow me to see logic better.

  - **Link:** LARP 3.0 Download - larp.exe

- Making progress all I got to do now is find light intensity levels to decide if light detect = True or False



- Completed pseudocode (might have to tweak values

PROJECT LOG - 31/08/25

- Today I plan to desk check my algorithm to make sure it works
    - Will do this by desk checking each routine (begin statements) and combining up with a conclusion if I should start development

    - After completing all desk checks I have concluded that my algorithm works fine all that I have to do now is create a structure chart to get a better understanding of how data is passed through modules

- After that I plan to start coding and wiring my board (after I get the correct cables

PROJECT LOG - 1/09/25

- I plan on completing the structure chart today so I can start actually programming (best or worst part of this task)
    - I began by scoping my projects structure and how data is passed between different modules/subroutines for the projects codebase

- Using draw.io to create the structure.
    - I relooked at the pseudocode to gain an understanding of how to approach this.

- Course specs notes:

Structure charts

Structure charts represent a system by showing the separate subroutines that make up the system and their relationship to each other.

Symbols

An empty circle is used to indicate data movement between subroutines (usually passed as parameters).

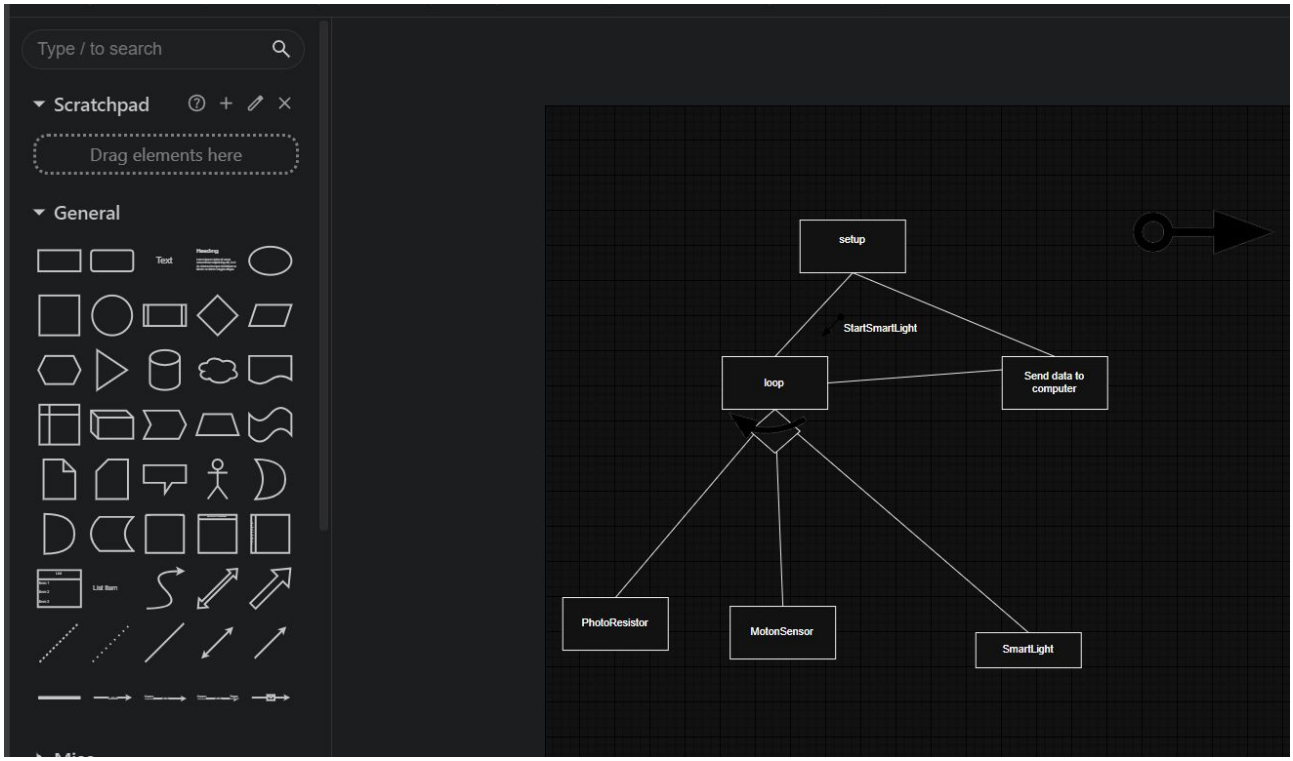
A filled circle is used to indicate a flag or control variable that is passed between subroutines.

A decision (ie optional execution of a subroutine) is indicated by use of a small diamond at the intersection of the connecting lines between subroutines that are called as the result of a binary or multi-way selection. Alternatively, the diamond may appear on a single connecting line if calling that subroutine is optional. In the diagram shown, a report may not necessarily be produced each time a book is returned.

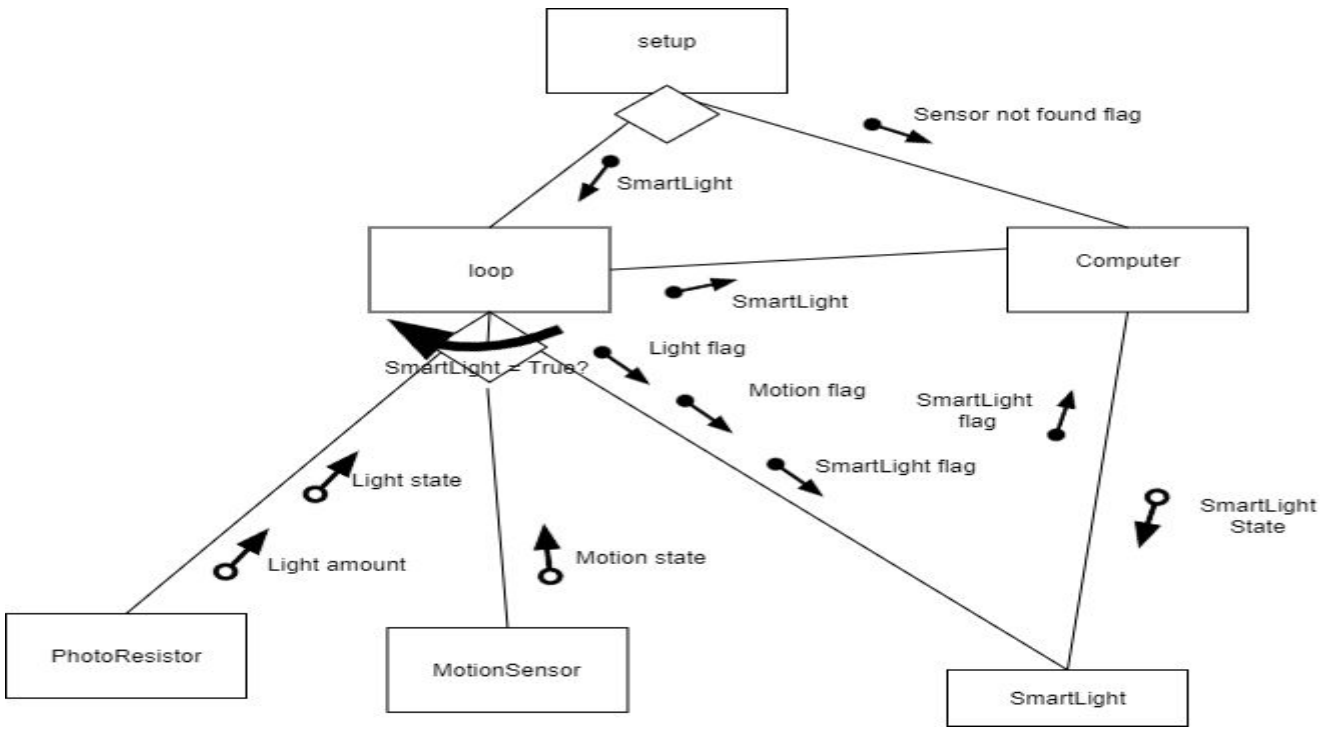
Repetition (execution of a particular subroutine or set of subroutines multiple times) is shown by a curved arrow.

The following structure chart represents a library system.

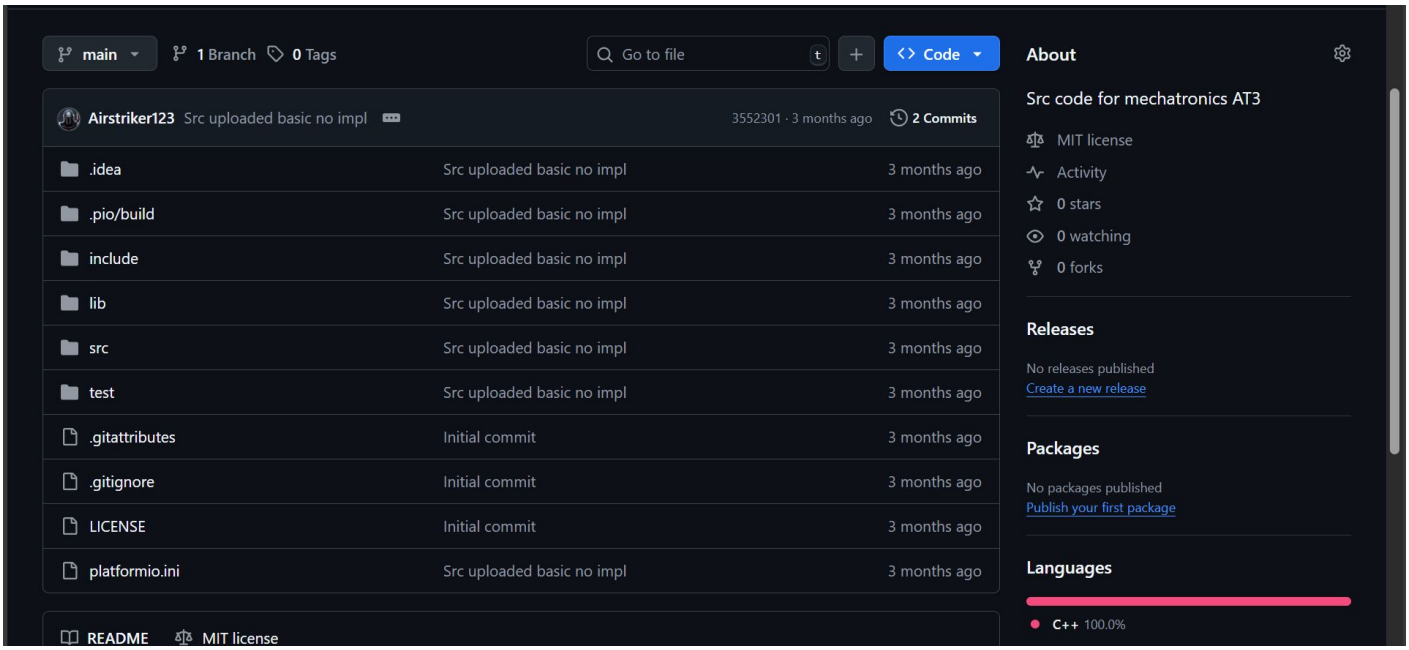
- Progress:
  - Finished base structure now I need to determine flow of data and how the program interacts with it

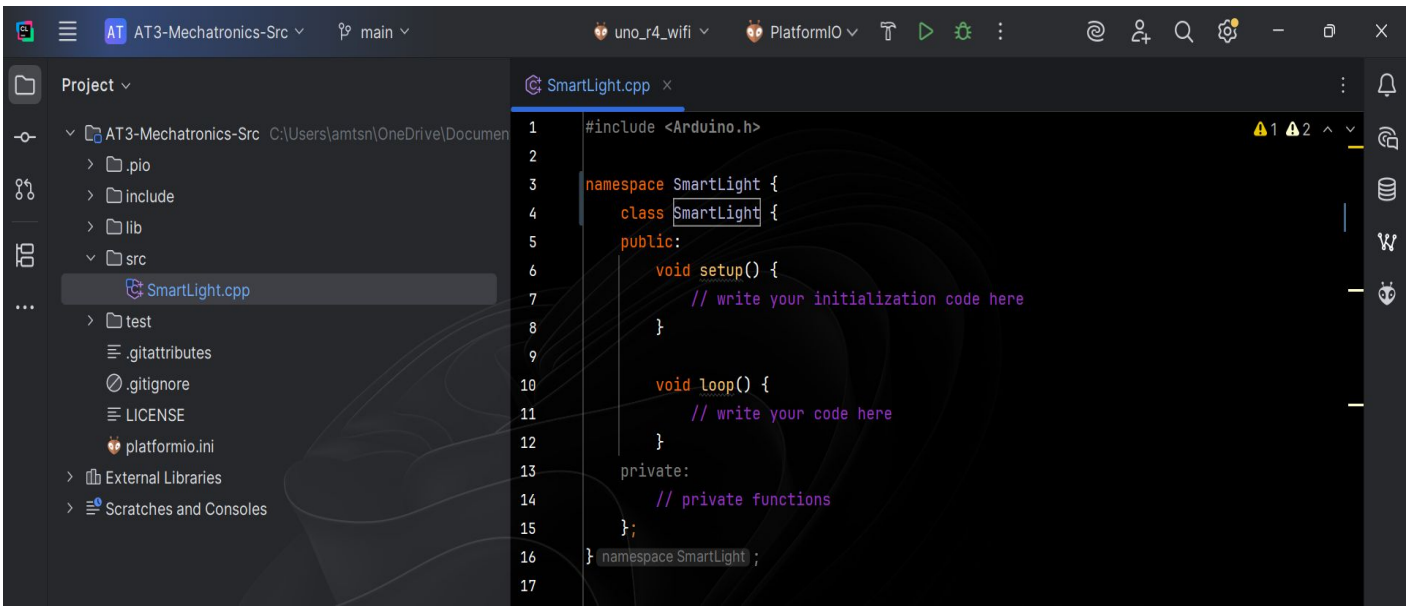


- Completed the structure chart by assessing the data flow and conditions (flags) to pass through modules
  - My solution seems to have a solid data structure



- Started coding using clion (I created a base template for coding.
- Uploaded to github for access to any device
  - Also incase I somehow lose it or project becomes corrupted.
  - Project is private for obvious reasons





- Completed most of code
  - Still Needs to be tested on real hardware.

PROJECT LOG - 2/09/25

- Completed project data dictionary
  - I began by looking at the course specifications to learn how nesa wants the format so I can replicate that in the correct form

- The documentation:

Data dictionary

A data dictionary provides a comprehensive description of each variable stored or referred to in a system. This commonly includes variable name, data type, format, size in bytes, number of characters to display the item including number of decimal places (if applicable), the purpose of each variable and a relevant example. Any validation rules applicable to the data item can also be included.

Details of records or arrays of records can be included in data dictionaries.

An extract of a data dictionary is shown.

Variable	Data type	Format for display	Size in bytes	Size for display	Description	Example	Validation
UserId	String	XXNNN	5	5	A primary key, uniquely identifies user. First 2 letters of surname followed by unique 3-digit identifier	PT173	First 2 characters letters, followed by last 3 characters digits
UserName	String	XX..XX	15	15	Username of employee	Kim	First letter is a capital letter
DOB	Date and Time	YYYY/MM/DD	4	10	Birth date of employee	1953/10/05	Valid date less than today
Times_Late	Integer	NNN	2	3	Number of times late to work this year	147	Integer between 0 and 999
PayRate	Floating Point	$NNN.NN	4	7	Hourly rate of pay	$124.37	Decimal greater than 20, less than 400
SocialClub	Boolean	X	1 bit	1	Y or N	N	
Departments	Array (string)		20 * number of departments	N/A	Names of departments in organisation	Administration Finance Marketing	From a drop-down list

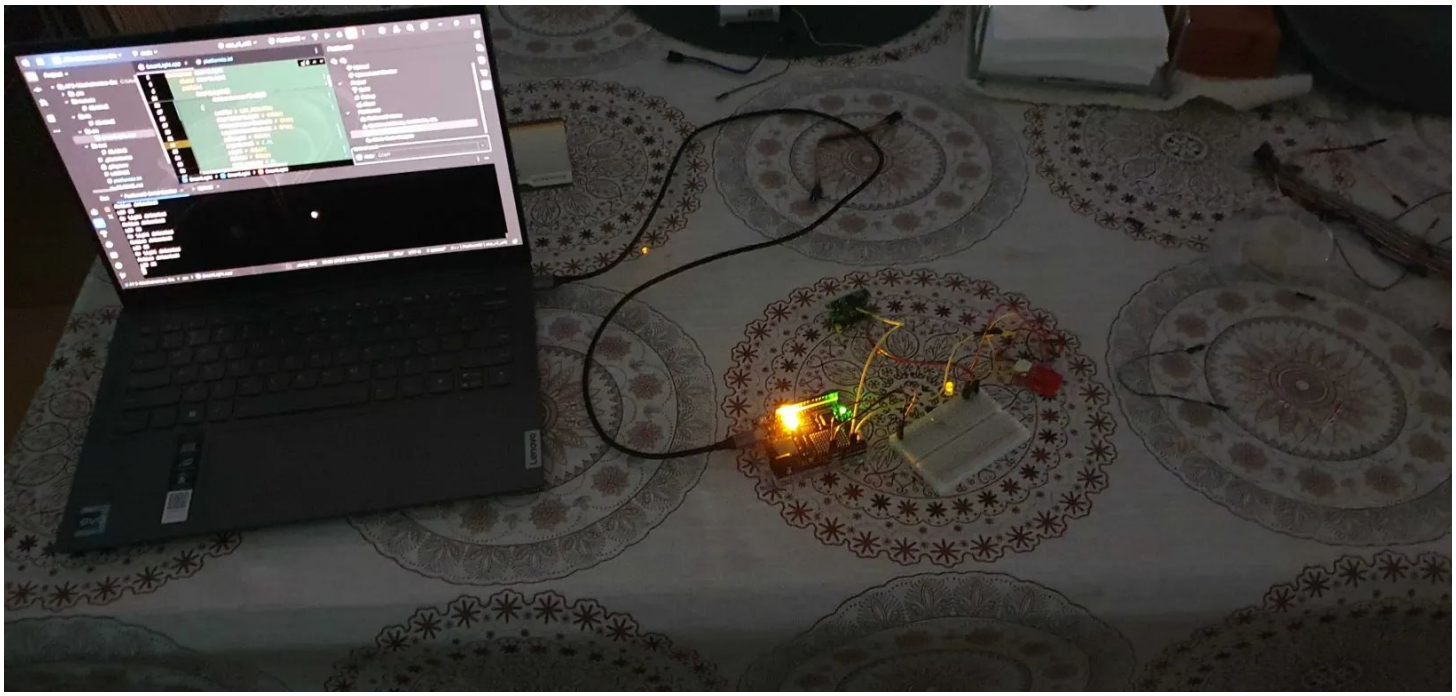
Note: a date and time data type is always stored as 32 bits (4 bytes) and can be displayed using different formats such as DD/MM/YYYY hh:mm:ss or YYYY/MM/DD

- Next day: I plan on getting the correct wires so I can test out my solution.
- Completed data dictionary

PROJECT LOG - 3/09/25

- Today I plan on getting cables and testing to see if my code works (I might wire the board in class)
    - I forgot my usb-c cable (no power to board ig)

- I started creating my board.



- Board was complex to make encountered many wiring errors and software errors
    - Such as computer could not upload code due to undetected usb
    - Other ide errors
    - Restricted device
    - Light going on when it not meant to

PROJECT LOG - 3/09/25

- Removed the motion sensor code and module from my project as it was redundant and not needed due to the use of light detector.
    - Sensor was also false detecting motion for no reason
        - Most likely hardware error as code was fine.
        - Removed algorithm from the code and edited the pseudo code.

- Uploaded code to board this time light was not working
    - Solution was wiring related as code was proven to work fine (it was trying to turn led light off when light detected but because location it still supplied electricity to the led light
    - Messed around with resistors to find best one (some were bad as brightness was too low.
- Completed the major project

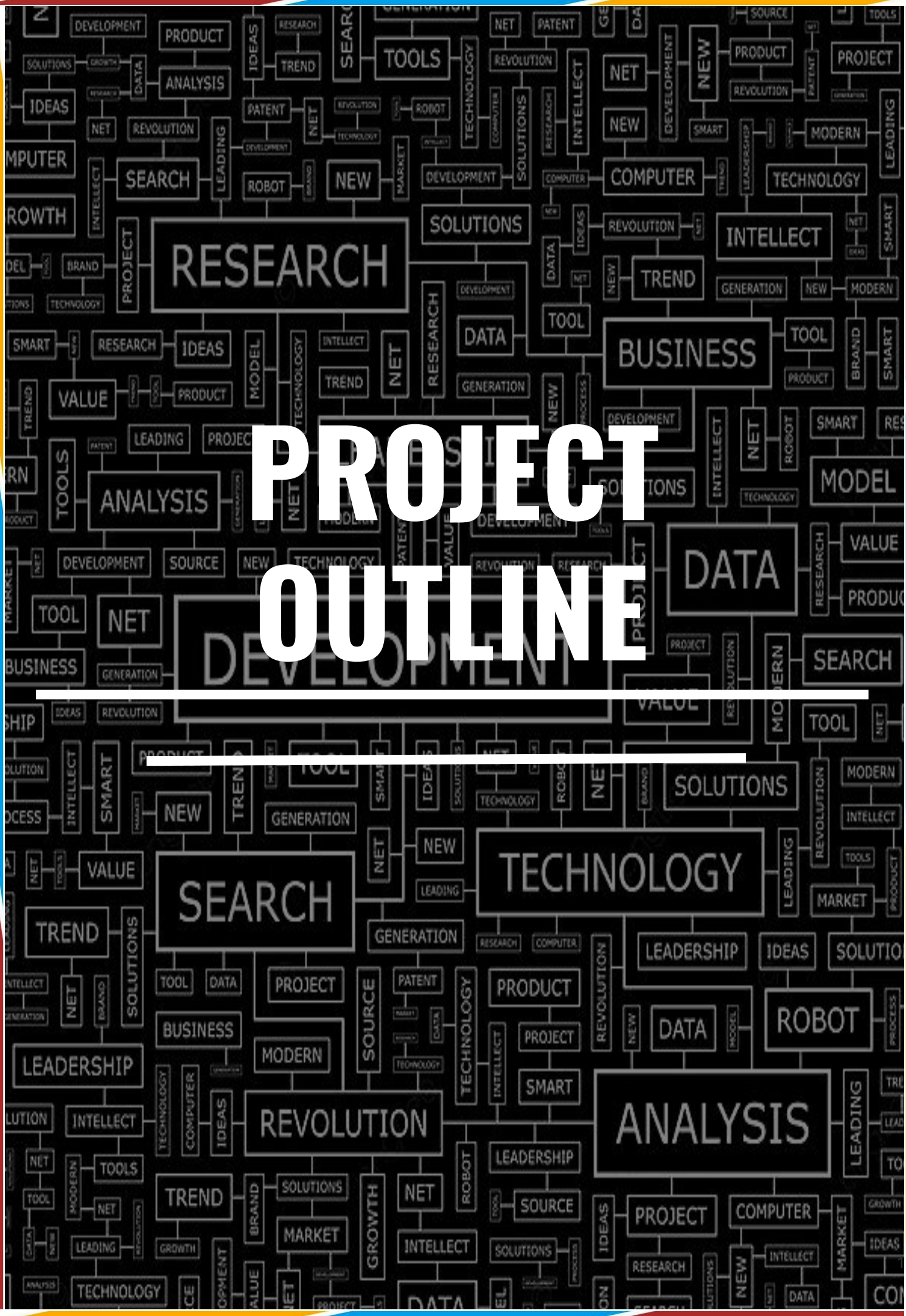
- Now I need to document the progress I made and determine for any final submission.

- Completed almost all project documentation

- Plan to make minor changes to report

- Completed project

- Realised the sensor in class was not working (It was working at home) for an unknown reason but jeff told me to submit and I will be fine.


PROJECT
OUTLINE

Statement of intent

What I intend to make:

I intend to make an eco-friendly smart home lighting system using an Arduino and sensors. The system will automatically switch lights on or off based on room occupancy and natural light levels, ensuring energy is not wasted.

- If I have spare time I might integrate wifi remote access to system via phone/laptop to control the system.

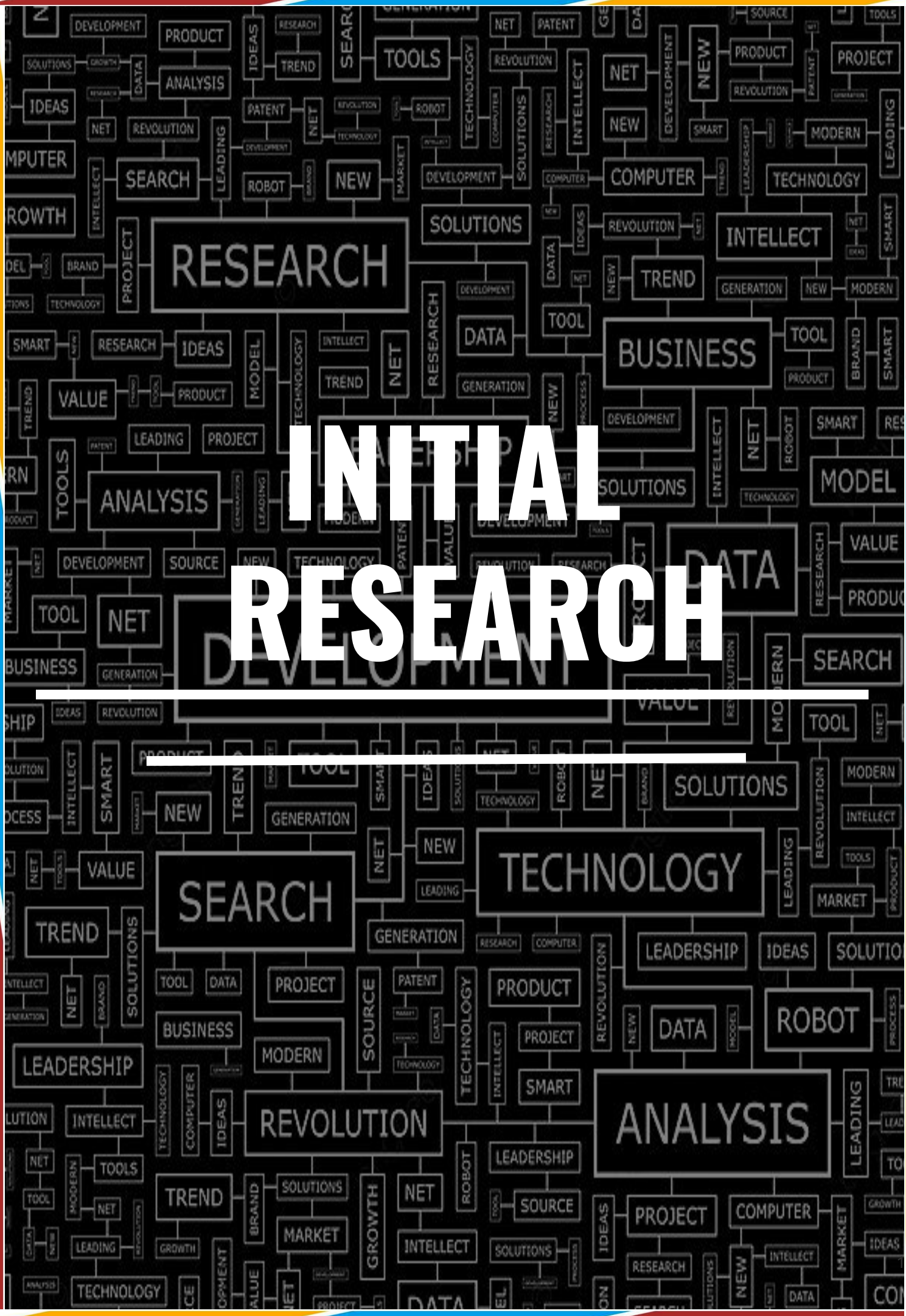
Why I intend to make it:

I want to create a project that improves quality of life at home by reducing the need for manual switching, while also encouraging households to be more eco-conscious. This project will demonstrate how automation can save electricity, reduce costs, and make homes smarter and more efficient.

How it is related to a real life problem:

A common issue in many households is leaving lights on unnecessarily, which increases electricity bills and contributes to environmental impacts like higher carbon emissions. By automating the control of lights using sensors, this project addresses the real-world problem of energy wastage while promoting sustainable living.

- This is from personal experience


INITIAL
RESEARCH

Learning the software

When it comes to programming the fastest way to learn it by looking at example projects.

- **I needed to get more familiar with the arduino platform such as the file structure and method of coding.**
- **This was done by examining github projects on the platform**
- **Since I was using platformIO for development the setup was slightly different to the Arduino ide.**
- **Source:** [platformio/platformio-examples: PlatformIO Project Examples]

Why I used platformIO:

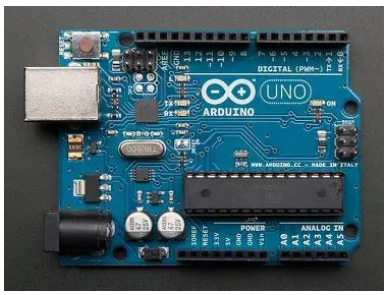
PlatformIO Core (CLI) is a unique, developed-from-scratch build system that removes the usual pain of software integration, packaging, and library dependencies that developers encounter when they move beyond the bounds of a specific SDK or example embedded application. It can be used with a variety of code development environments and allows easy integration with numerous cloud platforms and web services feeds. The user experiences no barriers to getting started quickly: **no license fees, no legal contracts**. The user maintains full flexibility of the build environment because the tools are open source and permissively licensed (no permission needed to modify them, and no requirement to share changes.)

Problematic

- The main problem which repulses people from the embedded world is a complicated process to setup development software for a specific MCU/board: toolchains, proprietary vendor's IDE (which sometimes isn't free) and what is more, to get a computer with OS where that software is supported.
- Multiple hardware platforms (MCUs, boards) require different toolchains, IDEs, etc, and, respectively, spending time on learning new development environments.
- Finding proper libraries and code samples showing how to use popular sensors, actuators, etc.
- Sharing embedded projects between team members, regardless of an operating system they prefer to work with.

Components required

Arduino Uno:



Purpose: The “brain” of the system. Reads sensor values, makes decisions, and controls the light (LED).

Why needed: Handles input (sensors) + output (light) + logic (if/else rules).

PIR motion sensor (HC-SR501):

Purpose: Detects if a person is moving in the room.

Why needed: Ensures the light only turns on when someone is actually present, reducing wasted energy.

LDR (photoresistor/ or use module):

Purpose: Measures how bright/dark the environment is.

Why needed: Prevents the light from turning on when there’s already enough daylight (saves electricity).

10 kΩ resistor (for the LDR voltage divider)

Purpose: Creates a readable voltage output when combined with the LDR.

Why needed: Arduino analog pins can’t read resistance directly — they need voltage.

LED (simulates the room light)

Purpose: Acts as the “smart light” in testing/simulation.

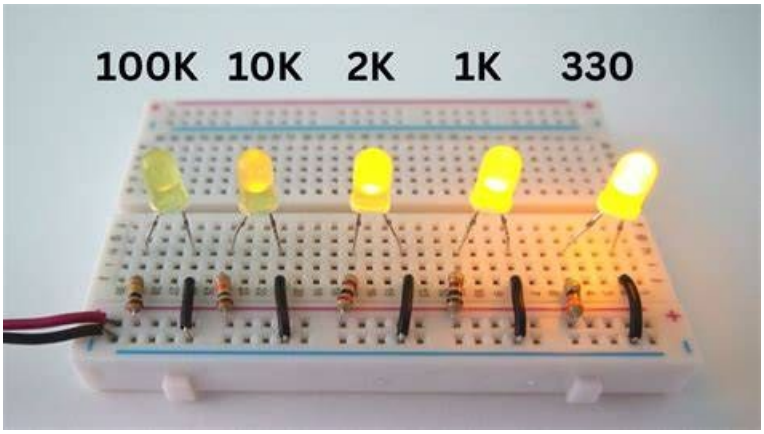
Why needed: Proof of concept

- later can be replaced by a relay or actual lamp.

220 Ω resistor (current-limit for the LED):

purpose: Protects the LED from burning out by limiting current.

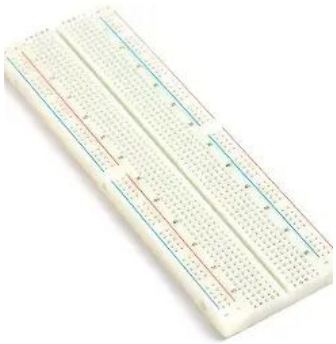
Why needed: Keeps the LED safe and stable.



Breadboard:

Purpose: Provides a reusable platform to build the circuit without soldering.

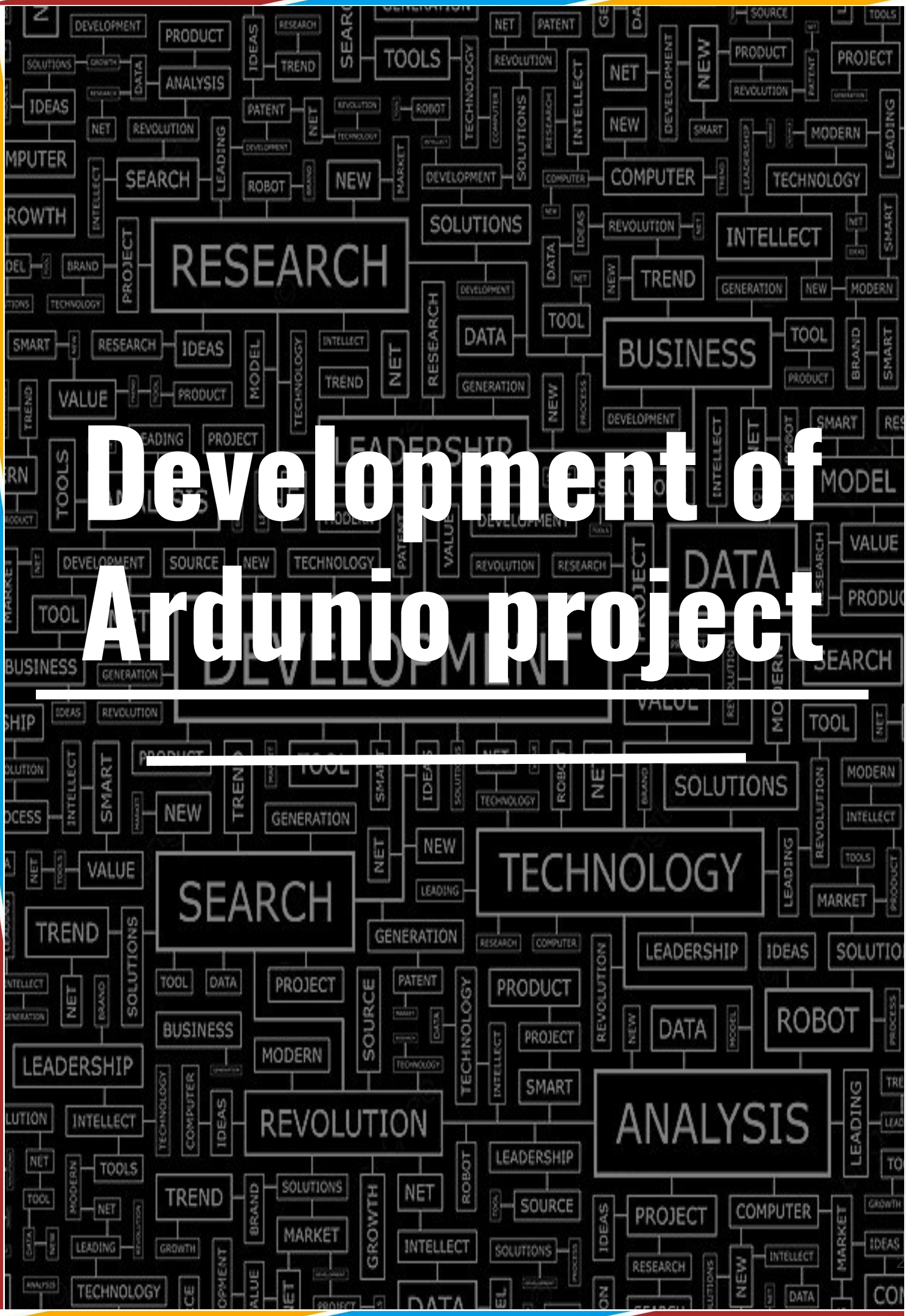
Why needed: Makes prototyping easier, allows components (Arduino, sensors, resistors, LED) to be connected quickly and rearranged if changes are needed.



Jumper wires:

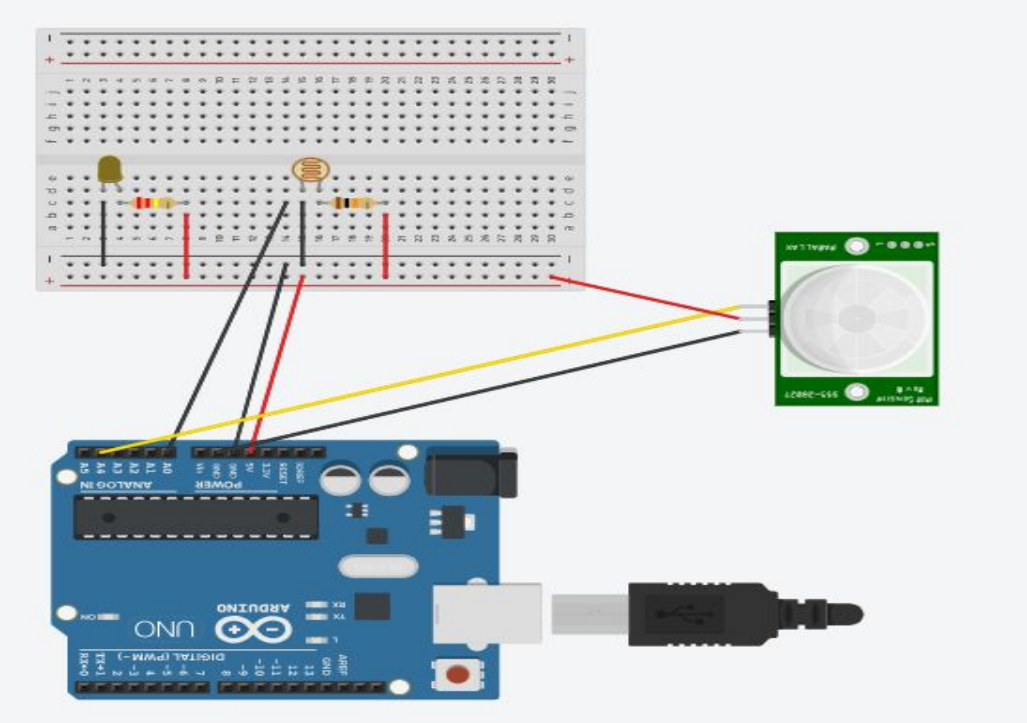
Purpose: Carry electrical signals between the Arduino, sensors, and other components.

Why needed: Essential for connecting the whole circuit together on the breadboard. Without them, the Arduino wouldn't be able to talk to the sensors or control the LED.

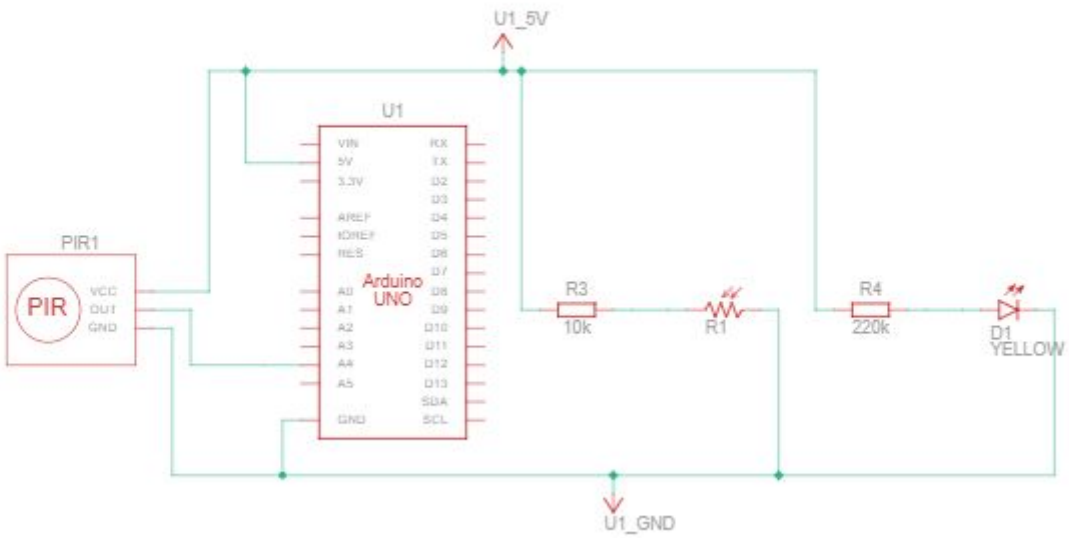

Development of
Ardunio project

Initial design concept

Circuit design (no annotations for clean view):



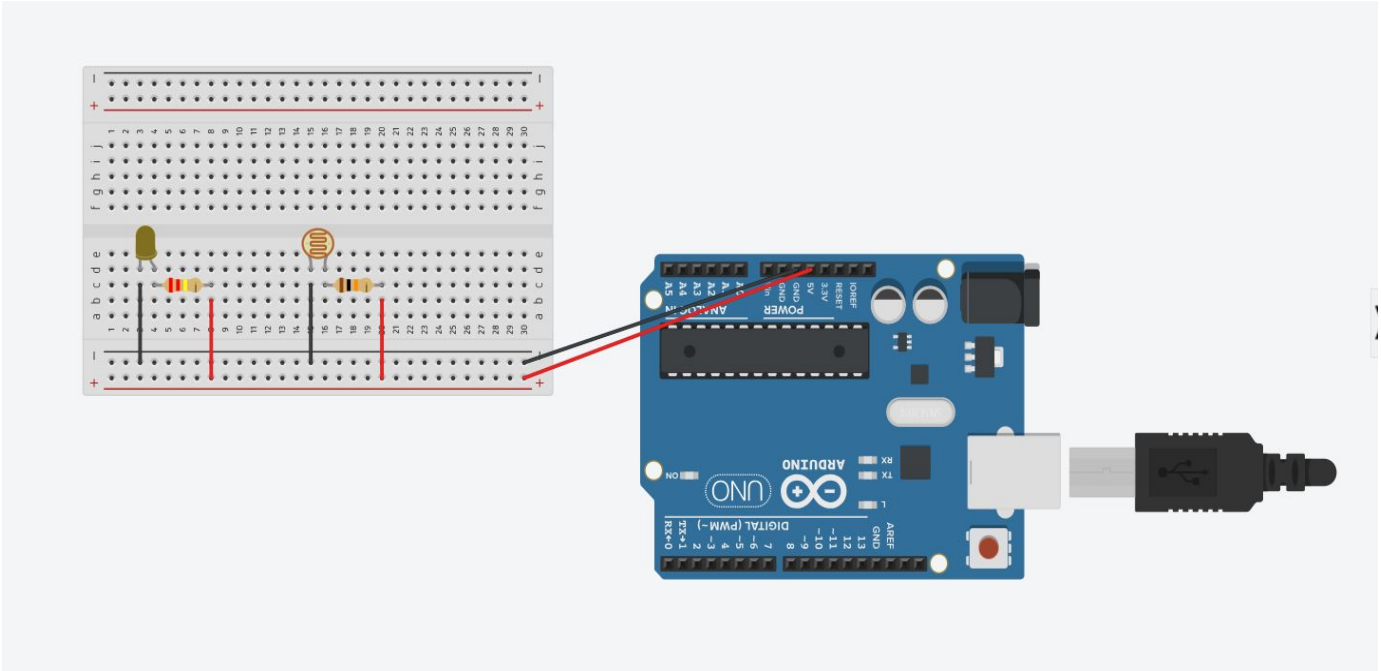
Schematic view:



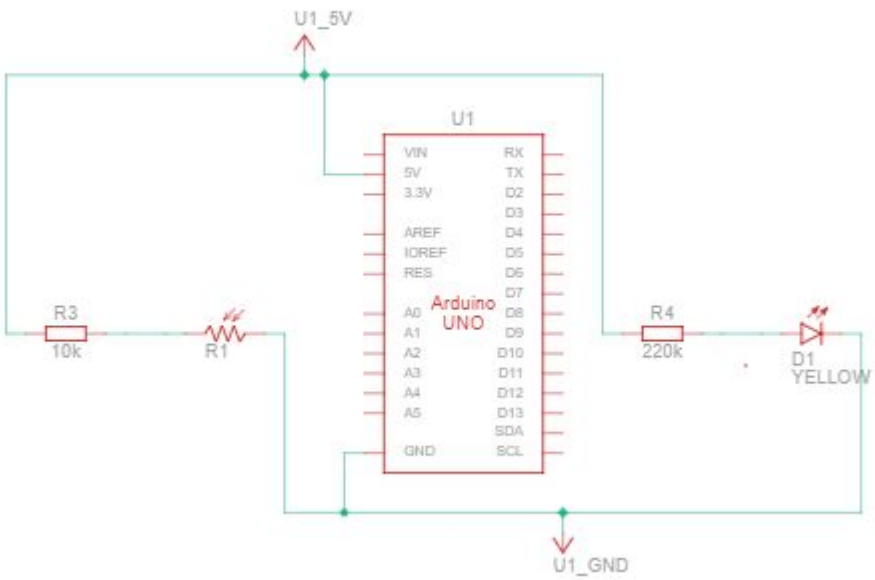
[Link of design with annotations (click me)]

Final design

Circuit design (no annotations for clean view):



Schematic view:



[Link of design with annotations (click me)]

Functionality concept

⚠️ NOTE: this is the a idea of how the project will function to help me determine how to develop solutions

This project is designed as a **smart lighting system** that responds to both **motion** and **ambient light conditions**.

1. **Light Detection (LDR – Light Dependent Resistor) -> using module from kit:**
    - The LDR continuously measures the ambient light level.
    - If the environment is bright (day time), the system keeps the LED off to save energy.
    - If the environment is dark (night time), the system allows the PIR sensor to control the LED.
2. **Motion Detection (PIR Sensor – the white ball <- best description):**
    - The PIR sensor detects human movement within its range.
    - When motion is detected **and** it is dark, the PIR sends a HIGH signal to the Arduino.
    - This triggers the LED to turn on, simulating an **automatic night light**.
3. **Output Control (LED):**
    - The LED acts as the visible output of the system.
    - It only lights up when **motion is detected** and **light levels are low**.
    - Once motion stops, the LED turns off after a short delay (configurable in code).
4. **Arduino Uno – Central Control:**
    - The Arduino reads input values from both sensors (analog input from the LDR, digital input from the PIR).
    - It processes the data using logic conditions (if PIR detects motion AND LDR < threshold → turn LED ON).
    - The Arduino then outputs a HIGH or LOW signal to the LED, controlling its state.

Possible future developments

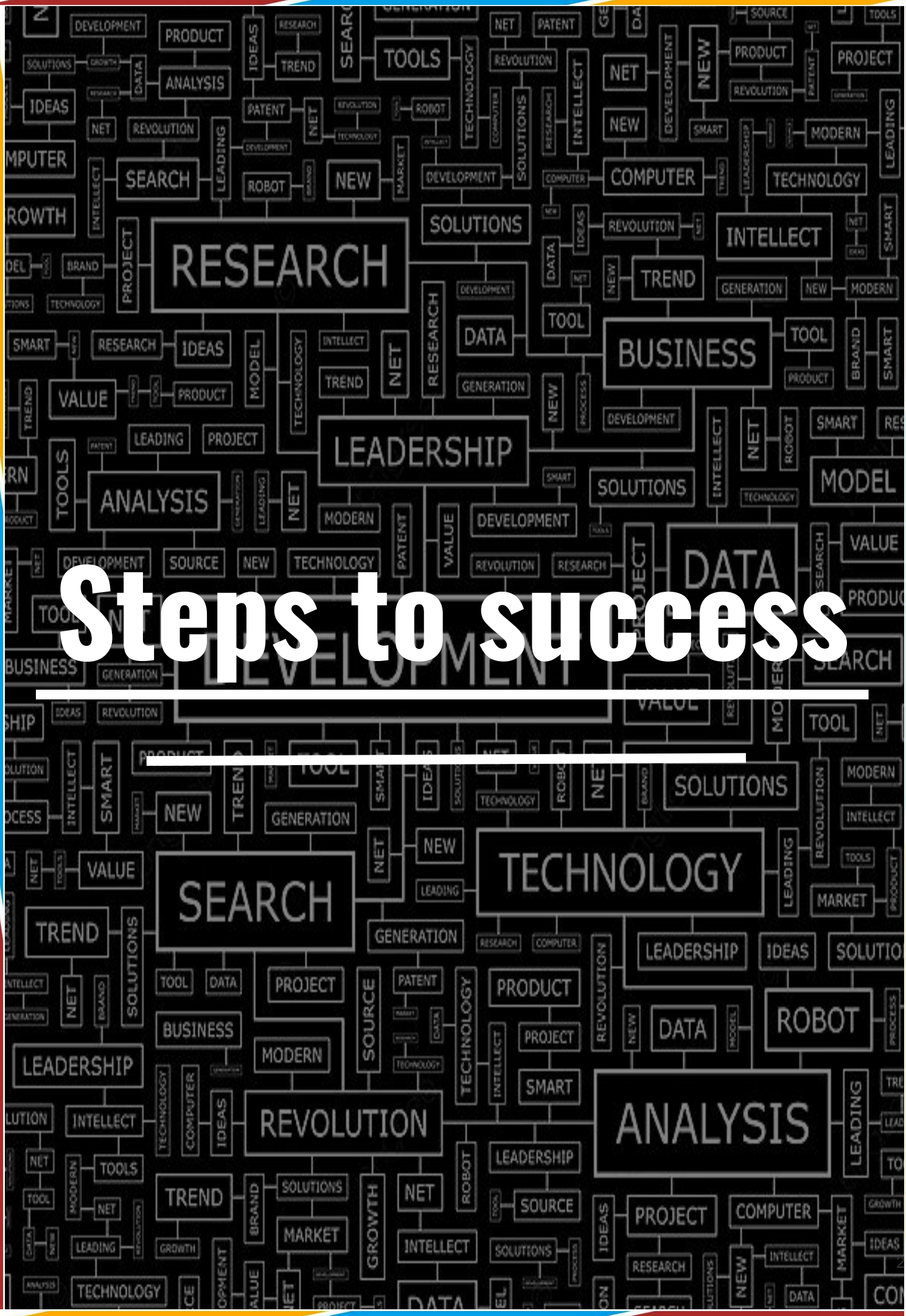
While the current prototype demonstrates a simple smart lighting system using motion and light sensors, there many ways it could be expanded in the future to make it more intelligent, user-friendly, and connected.

1. **Remote Control via PC or Smartphone (if I have time I may try to attempt this feature)**
    - Integrate a Wi-Fi module (e.g., ESP8266/ESP32) or Bluetooth (e.g., HC-05).
    - Allow users to monitor and control the system from a web interface or mobile app.
    - Example: Turning the lights on/off manually, viewing motion logs, or adjusting brightness thresholds remotely.
2. **Mobile Application Integration**
    - A dedicated app (iOS/Android) could provide a dashboard to configure sensitivity, light duration, or schedule times.
    - Push notifications could alert the user if unexpected motion is detected at night (security application).
3. **Voice Control / Smart Assistant Integration**
    - Add a **microphone sensor** or integrate with smart home assistants (Amazon Alexa, Google Home, Siri).
    - Users could issue voice commands such as *“Turn on the light”* or *“Set night mode”*.
    - This would require combining Arduino hardware with cloud services or APIs.
4. **Hardware Expansion**
    - Replace the single LED with a **relay module** to control actual home lighting.

Project limitations

Although the prototype demonstrates the concept of a smart motion-activated light, there are several limitations that I encountered which affected my projects functionality, usability, and scalability:

1. **Hardware Limitations**
    - The prototype only uses an **LED** as the output, which doesn't represent the brightness or power of a real household light.
    - **Arduino Uno** has limited memory and processing power, restricting the complexity of code and future add-ons.
    - The system relies on **wired jumper connections** on a breadboard, which are not durable or suitable for long-term use.
2. **Sensor Accuracy**
    - **PIR sensor** may produce false triggers due to environmental heat sources (sunlight, heaters, pets).
    - **LDR** provides only a basic light level reading and is sensitive to placement — results can vary depending on shadows or direct light.
3. **Power Supply Constraints**
    - project currently depends on **USB/Arduino 5V power**, which is fine for testing but not suitable for continuous operation or real-world installations.
    - No backup power or battery option is included, so the system stops working if disconnected.
4. **Software / Logic Limitations**
    - The system can only perform **basic ON/OFF control** — there is no dimming, scheduling, or remote override.
    - Delay times and thresholds need to be manually coded, instead of being adjustable through a user interface.
5. **Practicality / Real-World Application**
    - The prototype is not weatherproof, meaning it cannot currently be used outdoors.
    - It is not integrated into existing smart home systems or electrical wiring, limiting it to a demonstration model.
    - The design lacks an enclosure, so wires and components are exposed and fragile.


Steps to success

Requirements definition for given real world problem

Clearly identify the real-world problem:

People waste electricity when outdoor/indoor lights are left on unnecessarily. A system is needed to automatically switch lights on only when motion is detected in low light conditions.

Define project goals:

- Energy efficiency (light only when needed).
- Safety/security (motion-based activation at night).
- User-friendly (automatic, no manual switching).

List **constraints**:

- available hardware (Arduino Uno, LDR, PIR, LED)
- limited power supply
  - Currently powered by a usb
- breadboard prototyping.
  - Must make sure to know where to insert wires to avoid damage of hardware
  - Avoid misuse of jumper cables
  - Deciding which resistor to use (finding resistance is pain as colour is not easy to see)
- Software errors
  - Code not compiling
  - Code not uploading to board
  - Software just not working in general
  - Laptop not detecting board
- Hardware errors
  - Such as faulty modules (may encounter)
  - Incorrect choice of hardware and sensors
  - Trouble wiring
  - Jumper wire setup

Outline the project solution and how it solves the targeted problem

Project Concept:

Build a smart lighting prototype using Arduino Uno, PIR sensor (motion detection), and LDR (light detection).

How it solves the problem:

- Detects ambient light → prevents lights from turning on during the day which prevents unwanted and waste of electricity.

- Detects human motion → ensures lights only turn on when someone is present. (problem if someone is in room and motion is not detected light may turn off)
    - Solution: Add a timer about 5 minutes maybe and even if motion not detected don't turn light off instantly, instead maybe dim it.

- Automatic control → saves energy and reduces user effort which enhances quality of life reducing stress on users who may be anxious about this.

The expected outcome:

An LED light that automatically turns on when motion is detected in the dark, and turns off when not needed.

Data collection

LDR (Light Dependent Resistor):

- Collects data on surrounding brightness.
- Outputs an analog voltage (0–5V) to Arduino depending on resistance.
- Arduino uses a threshold value to decide if it is “day” or “night.”

PIR Sensor (HC-SR501):

- Collects data on infrared changes (motion).
- Outputs HIGH (1) when motion is detected, LOW (0) otherwise.
- Arduino checks PIR signal only when it’s dark (from LDR data).

Data Integration:

- Arduino processes both inputs:
  - Example logic concept:

```
IF LDR <= threshold AND PIR == HIGH THEN:
        Turn on LED
ELSE
        Turn off LED
```

Why it must be collected:

sensor data ensures the project behaves intelligently in real-world conditions. The LDR sensor detects the light threshold in the environment to determine if led light should be enabled, and motion sensor can also act as a security measure to determine if there's someone in a room.

Algorithm design And desk checks

⚠️ **NOTE: Final project does not use all of the algorithm design (the motion sensor parts) due to changes in plan however design process was left untouched to demonstrate my plan process**

- **Motion sensor pseudocode was not used in final project as I realised it was redundant for my project**

⚠️ **NOTE: Pseudocode includes code comments to help me understand processes that need to be raw english for a better understanding of algorithm (these act as annotations)**

Pseudocode

```
LEDLight = off
LightLevel = 0
Light = False
Motion = False
StartSmartLight = False

BEGIN PhotoResistor
    \\ process to request light from sensor and determine if light is on or off
    LightLevel = read LightLevel from sensor  \\ sensor will return light intensity

    \\ determine if to mark as light detected or not to avoid false positive
    IF LightLevel > 150 THEN
       Display "Sensor is detecting light!"
       RETURN  Light = True
    ELSE
       Display "No light detected"
       RETURN  Light = False
    ENDIF
END PhotoResistor


BEGIN MotionSensor
    \\ process to perform motion check
     Motion = read from sensor
     IF Motion THEN
        Display "Motion detected!"
        RETURN  Motion = True
     ELSE
        Display "No motion detected"
        RETURN  Motion = False
    ENDIF
END MotionSensor
```

```
\\ Ardunio setup process only executes once
BEGIN setup
    \\ safety checks to check if product sensors are working
    MotionSensorStatus = read from sensor
    LightSensorStatus = read from sensor
    Sensor1 = False
    Sensor2 = False

    IF MotionSensorStatus = True THEN
       Sensor1 = True
       Display "Motion sensor detected!"
    ELSE
       Display "Motion sensor not detected!"
    ENDIF

    IF LightSensorStatus = True THEN
       Sensor2 = True
       Display "Light sensor detected!"
    ELSE
       Display "Light sensor not detected!"
    ENDIF

    IF Sensor1 AND Sensor2 = True THEN
       Display "Smart Light is now active!"
       RETURN  StartSmartLight = True
    ENDIF
END setup



\\ process to perform real time checks if any change in environment
BEGIN loop
   IF StartSmartLight = True THEN \\ only start if all sensors work
     IF Motion AND Light = True THEN
        Display " LED light on!"
        LEDLight = On
     ELSE
        Display "LED light off"
        LEDLight = off
     ENDIF
   ENDIF
END Loop
```

Desk checks

Desk check for Photoresistor

Given values: `80 , 150 , 200`

LightLevel	`LightLevel > 150`	Output	Light
80	False	“No light detected”	False
150	False	“No light detected”	False
200	True	“Sensor is detecting light”!	True

Desk check for Motion Sensor

Given values: . LOW, HIGH

Motion (from sensor)	Output	motion
LOW	“No motion detected”	False
HIGH	“Motion detected!”	True

Notes: motion sensor may cause problems in my project so I might just remove it from final product to avoid unwanted conditions of led light.

Desk check for setup

Given values:

False, False
True , False
False, True
True , True

MotionSensorStatus	LightSensorStatus	Condition sensor1 (MotionSensorStatus = True)	Condition sensor2 (LightSensorStatus = True)	Condition (Sensor1 AND Sensor2 = True)	Output	StartSmartLight
False	False	False	False "	False	“Motion sensor not detected !Light sensor not detected !”	False
True	False	True	False"	False	“Motion sensor detected !Light sensor not detected !”	False
False	True	False"	True	False	“Motion sensor not detected !Light sensor detected !”	False

True	True	True	True	True "	“Motion sensor detected!Light sensor detected!Smart Light is now active!”	True

Desk Check for loop:

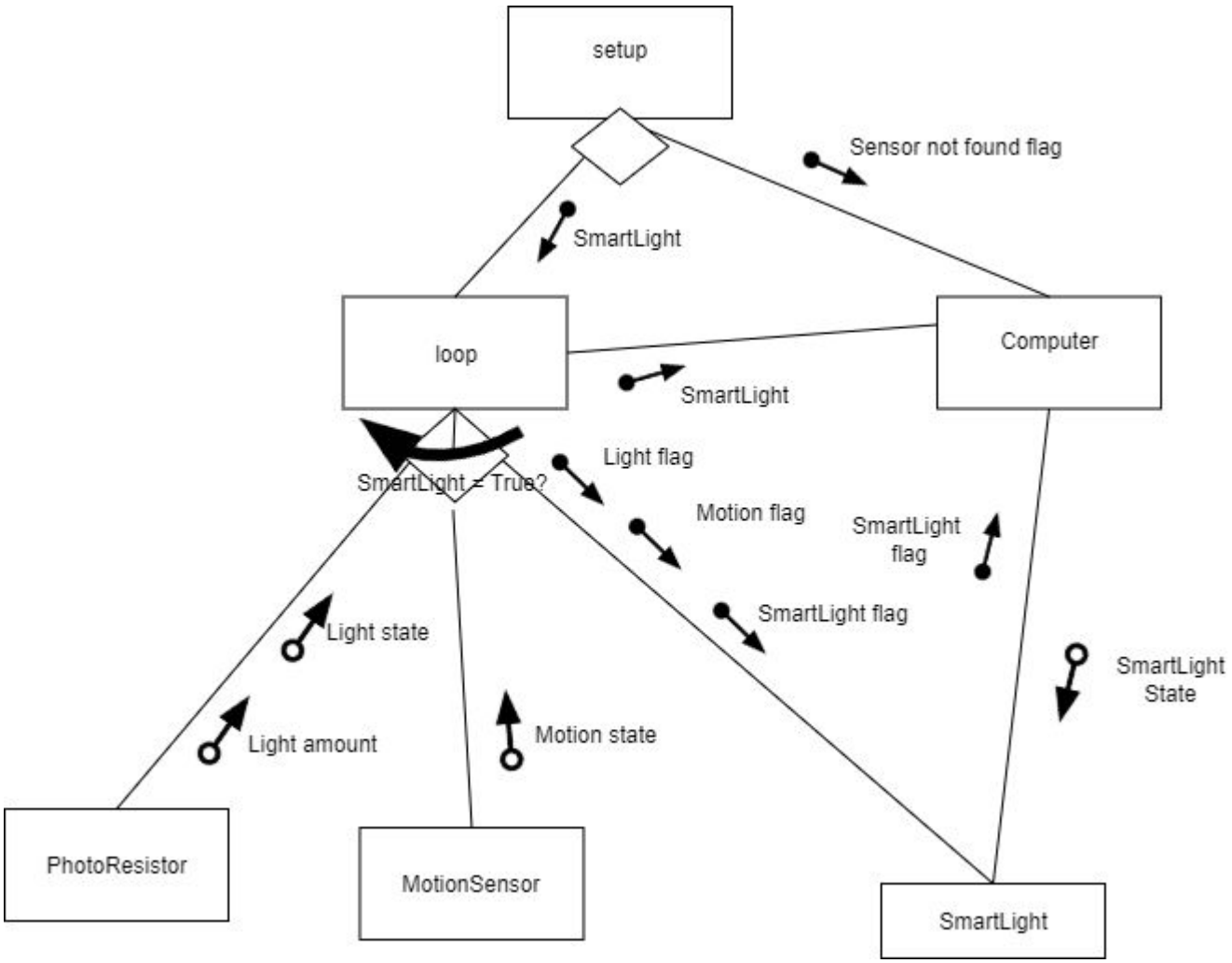
Given values:

False, True , True
True, True, True
True, True, False
True, False, True
True, False, False

StartSmartLight	Motion	Light	LEDLight	Output
False	True	True		
True	True	True	On	"Light is on"
True	True	False	Off	"Light is off"
True	False	True	Off	"Light is off"
True	False	False	Off	"Light is off"

Conclusion: All desk checks produced the expected outcome and my algorithm works as intended!

Structure chart



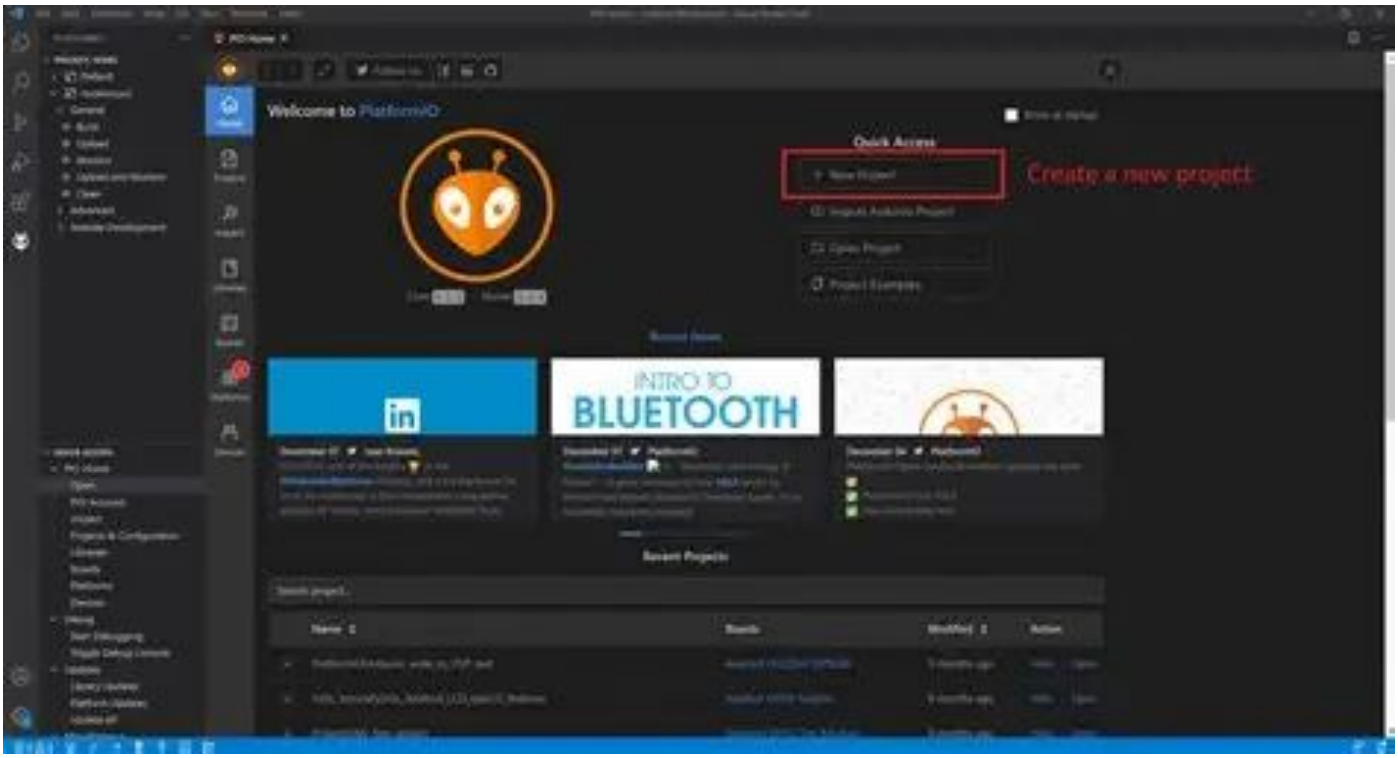

Development
environment setup

Choice of software

For this project instead of using the arduino ide I have decided to use alternative software which will increase my productivity as I am familiar with it. These choices are jetbrains Clion ide with the Platform io plugin.

Clion: Clion is an integrated development environment designed for c/c++ developers. Arduino is a subset of c++, which is why this ide will be a better choice then mechatronics.

Platformlo plugin for Clion: PlatformIO is an open-source ecosystem for embedded development. It allows working with various MCUs (such as ARM Cortex, AVR, MSP430), development boards, frameworks, and environments (Arduino, ESP-IDF, and many others).



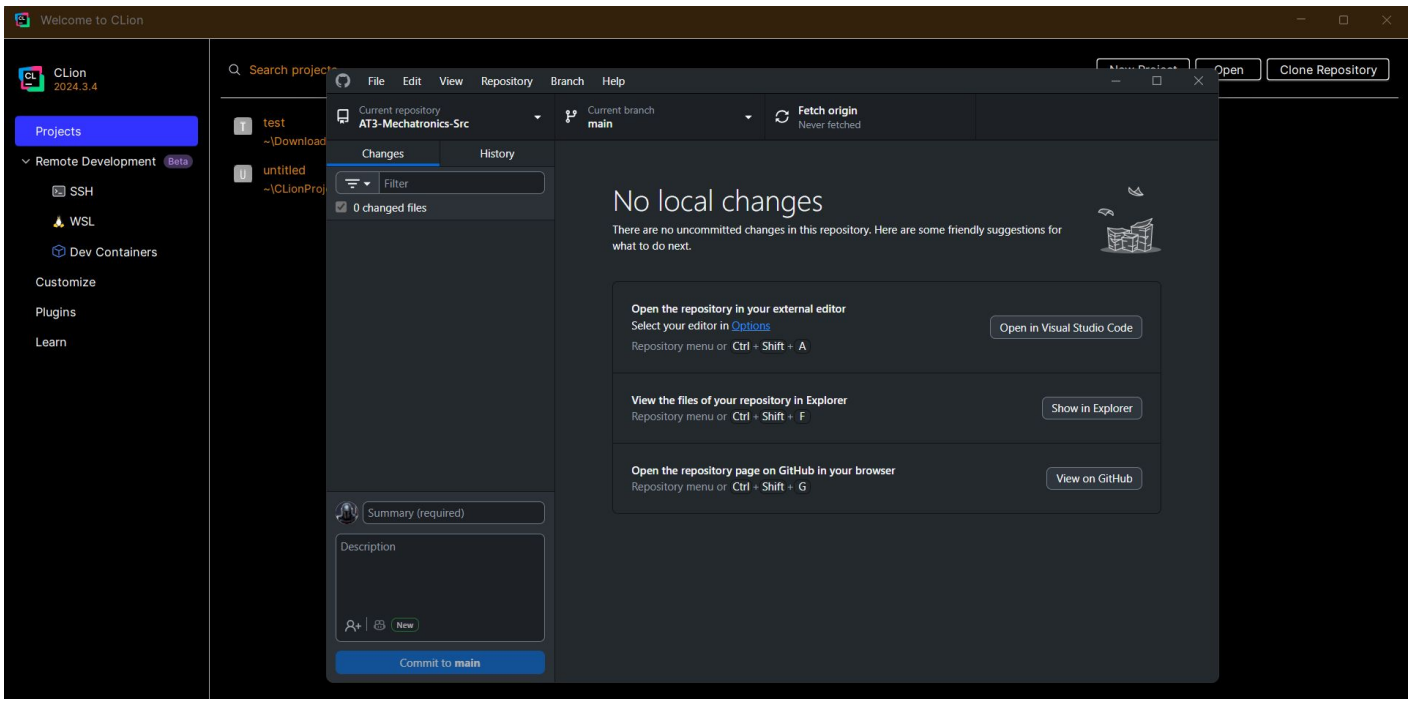

Developing
solutions

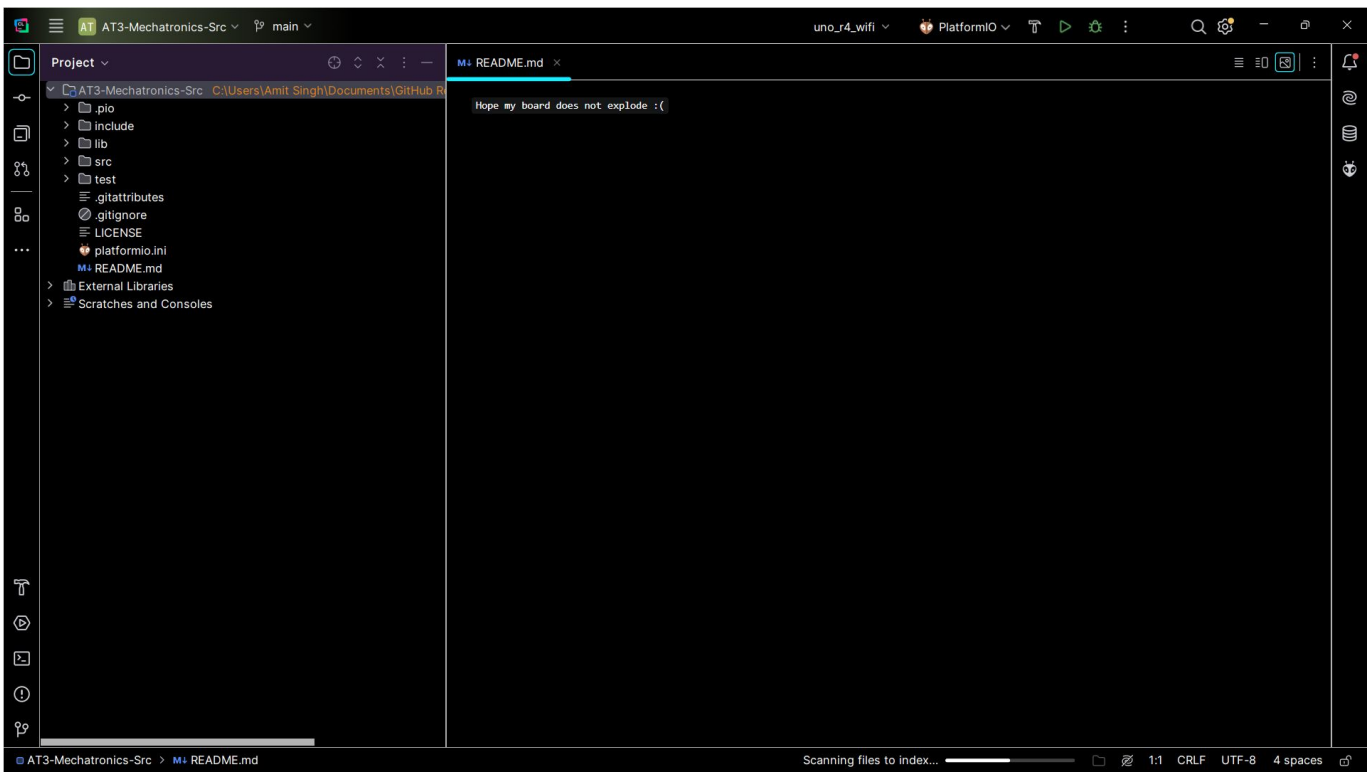
- Using github to keep a copy of the project incase any error occurs which may corrupt or cause me to lose progress on project.
    - Github repo link: **click me**
- I started looking at how other people were coding their solutions in github to understand how c++ works and it's difference to python.
    - On the way I found this very interesting and useful repo which can modules for all sensors
    - I plan to use this for reading sensor data while still coding conditions to turn on / off the light
        - Repo link: **click me**

#	Project	Component / Concept	Description
01	`01-photoresistor.ino`	Light Detection	Measures ambient light levels using a photoresistor
02	`02-thermistor.ino`	Temperature Sensing	Reads temperature using a thermistor with analog input
03	`03-moisture.ino`	Soil Moisture Sensor	Monitors soil moisture levels for plant care
04	`04-tilt_switch.ino`	Tilt Detection	Detects orientation changes using a tilt switch
05	`05-pir_motion_sensor.ino`	Passive Infrared Sensor	Detects motion using a PIR sensor
06	`06-ultrasonic.ino`	Distance Measurement	Measures distance using HC-SR04 ultrasonic sensor
07	`07-humiture_sensor.ino`	Temperature & Humidity	Reads temperature and humidity using DHT sensor
08	`08-mfrc522.ino`	RFID Reader	Reads RFID cards using MFRC522 module
09	`09-gy87.ino`	IMU Sensor (GY-87)	Advanced IMU with accelerometer, gyroscope, magnetometer, and barometric pressure

- **Coded some logic and added pseudocode as code comments to help me determine what to do (it acts as a todo list.**

- **Cloned the github repo on my home computer to work on**





- **Now I can begin coding as environment is setup (I think)**

- **Finished developing the sensor code to detect if light and motion**
- **Now I need to develop condition for loop to determine if light should be on or off and setup to determine if to proceed with scanning to avoid any potential risk of damage.**

- **Light sensor code:**

```
34
35          bool LightSensor()
36          /*
37
38  */
39          {
40              try {Serial.begin(9600);
41                  int LightLevel = analogRead( pinNumber: PhotoResistorpin);
42                  if (LightLevel > 150){
43                      // condition for light detection
44                      return Light = true;
45                      Serial.print("Sensor is detecting light!");
46                  }
47                  // if not above 150 return no light detected
48                  return Light = false;
49                  Serial.print("Sensor is not detecting light");
50                  // Wait for 1 second before the next loop iteration
51                  delay(1000);
52              }
53              catch (...) {
54                  // incase sensor fails to notify me to fix it!
55                  Serial.println("Light Sensor could not be detected");
56              }
57          }
```

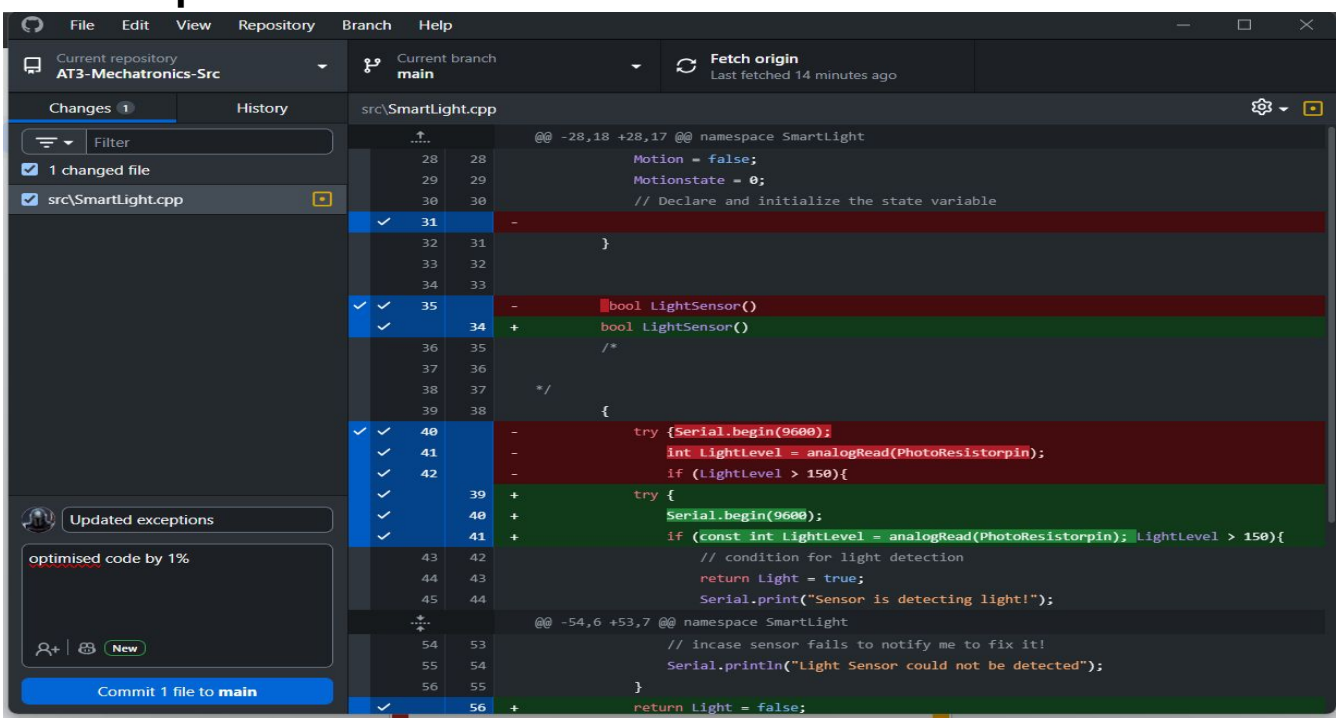
- **Motion sensor:**

```
  bool MotionSensor()
/*

 */
{
    try {
        delay(1000);
        Motionstate = digitalRead(MotionSensorPin);         // Read the state of the PIR sensor
        if (Motionstate == HIGH) {                 // If the PIR sensor detects movement (state = HIGH)
            return Motion = true;
            Serial.print("Motion is detecting motion!");
        }
        return Motion = false;
        Serial.println("Monitoring...");
    } catch (...) { //incase sensor fails to notify me to fix it!
        Serial.println("Motion Sensor could not be detected");
    }
}
```

- **Object oriented programming in c++ using classes:**
  - **Also a constructor**

```
5   namespace SmartLight
6   {
7       class SmartLight
8       {
9
10      public:
11
12          //class public variables to declare type
13          bool LEDLight;
14          double LightLevel;
15          bool Light;
16          bool Motion;
17          const int PhotoResistorpin; //define the pin
18          const int MotionSensorPin; //  define the pin
19          int Motionstate; //define pin = 0
20
21
22          SmartLight() //constructor
23          {
24              //predefine variables
25              LEDLight = false;
26              LightLevel = 0.0;
27              Light = false;
28              Motion = false;
29              Motionstate = 0;
30              // Declare and initialize the state variable
31
32          }
33
```

- **Completed almost half of the codebase**



- **Completed all of the code base**
  - **Loop code to determine if light is on/off**

```
    {
        if (StartSmartLight == true) { //only start if sensors work
            if (Light == true && Motion == true ) { //check if both sensors are detecting
                //turn light on if they are
                Serial.println("LED light on!");
                digitalWrite(LedPin, status: HIGH);
            }
            else { //turn light off if not
                Serial.println("LED light off!");
                digitalWrite(LedPin, status: LOW);
            }
        }
    }
};
namespace SmartLight ;
```

- **Completed setup code:**

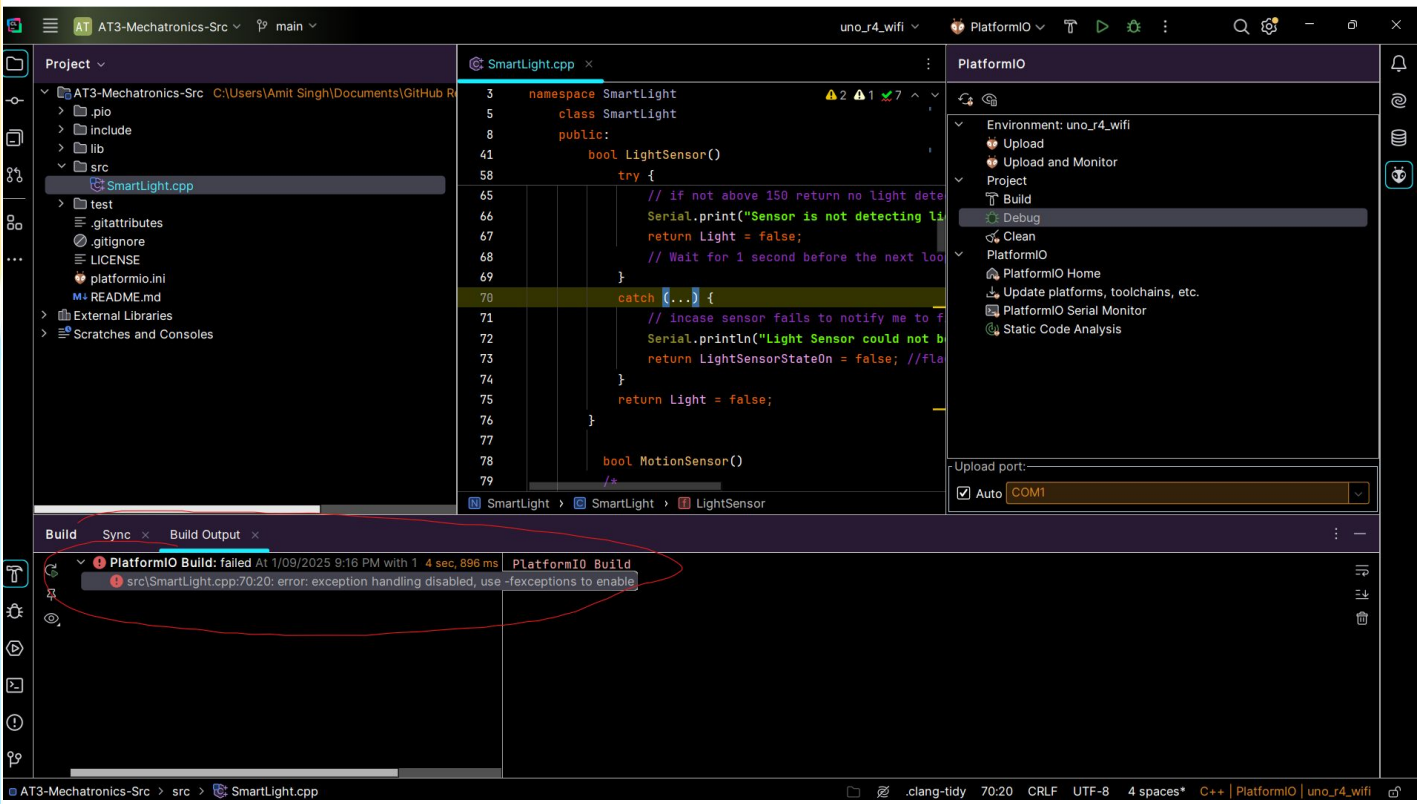
```
{
    pinMode(LedPin, pinMode: OUTPUT); //led location (readjust if needed)
    pinMode( pinNumber: MotionSensorPin, pinMode: INPUT);  // Set the PIR pin as an input
    Serial.begin(9600);
    Serial.println("initializing SmartLight setup!");

    if (MotionSensorStateOn == true) {
        Serial.println("Motion Sensor is on!");
    }
    else {
        Serial.println("Motion Sensor is off");
    }
    if (LightSensorStateOn == true) {
        Serial.println("Light Sensor is on!");
    }
    else {
        Serial.println("Light Sensor is off");
    }
    if (LightSensorStateOn == true && MotionSensorStateOn == true) {
        Serial.println("All sensors work!");
        Serial.println("Starting smartlight!");
        return StartSmartLight = true;
    }
}
```

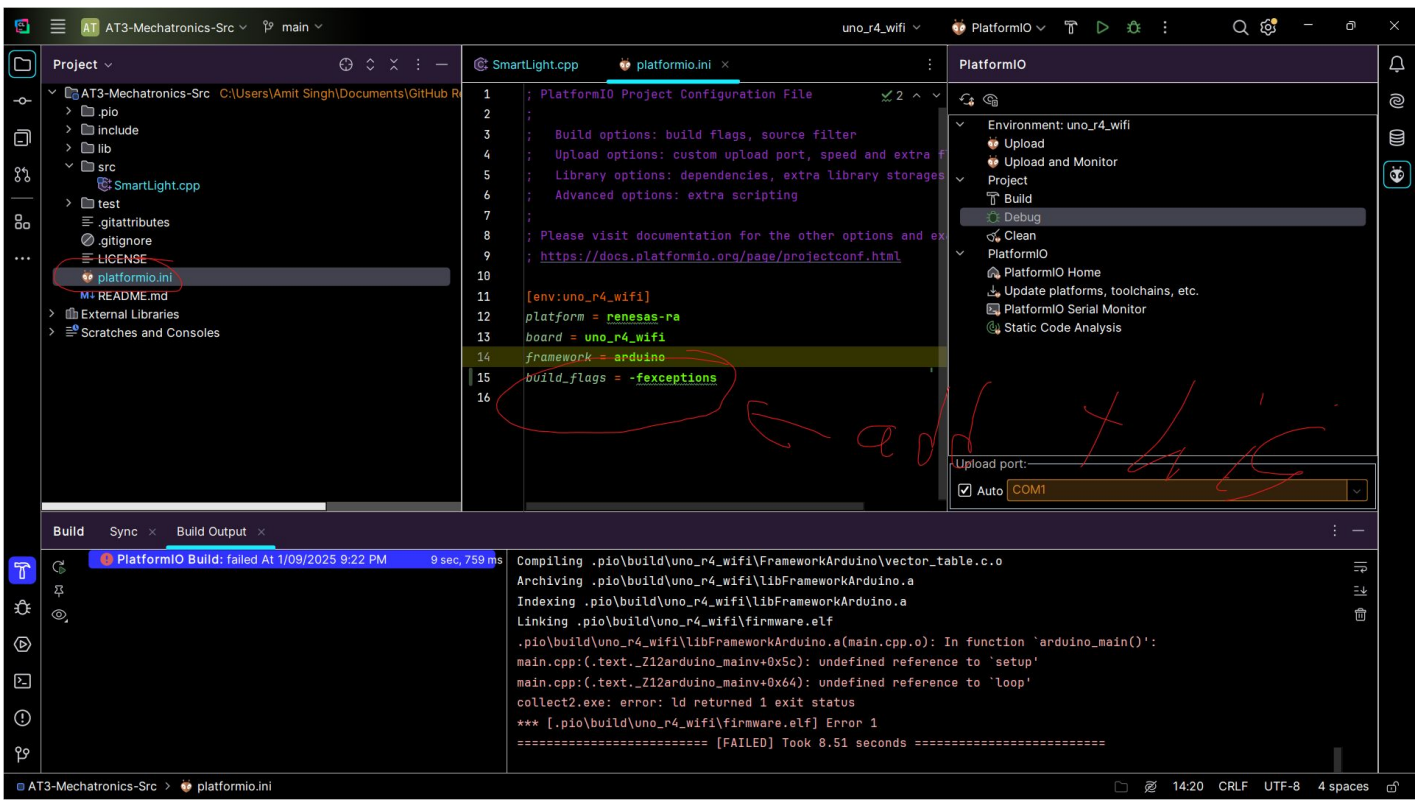
- **I think I completed the coding part**
  - **Still need to test to see if end result works or not**

Error type: build error

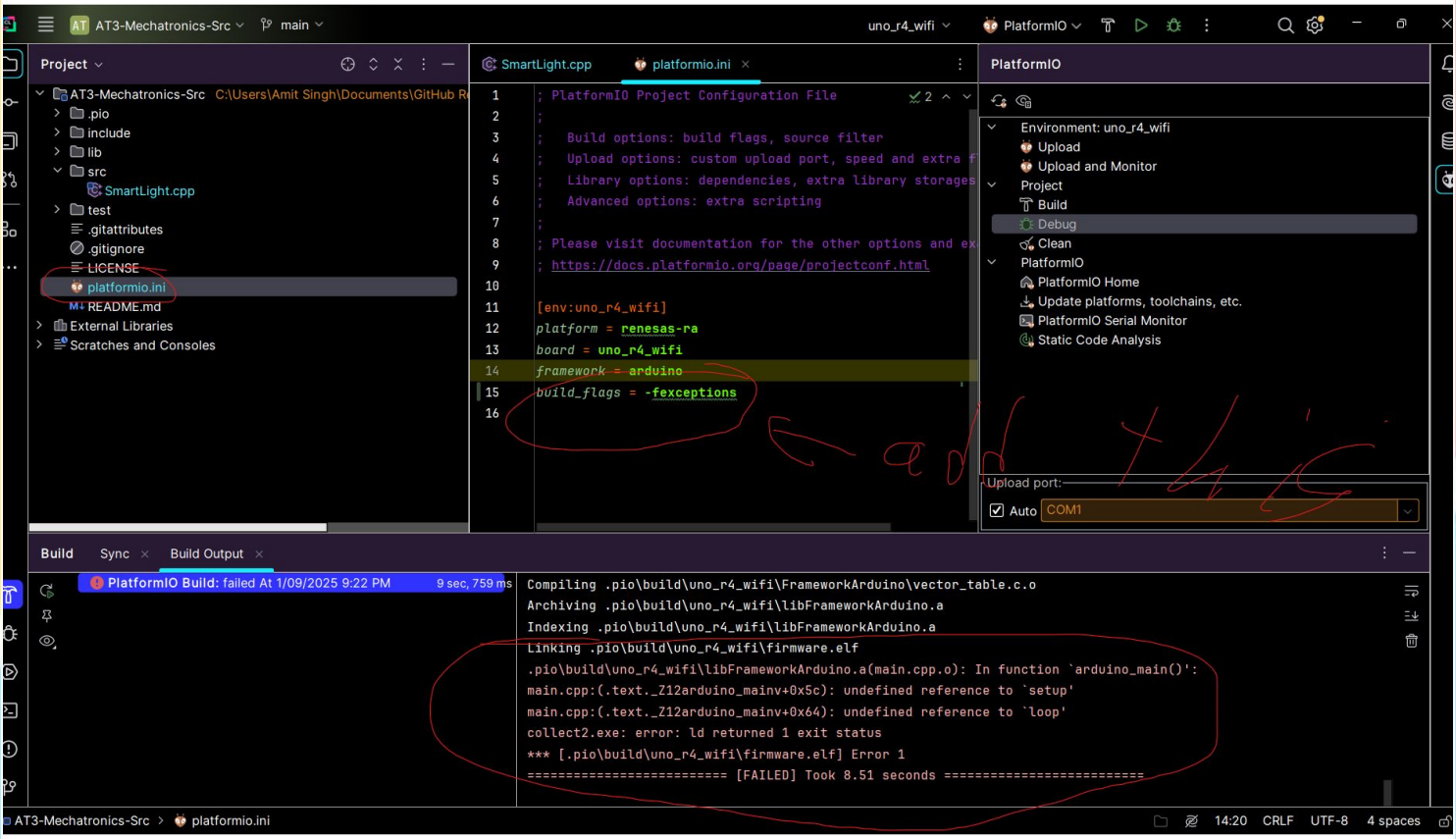
Occurred at: when building I received an error saying I can't use exceptions (this made me sad) and told me to use -fexceptions instead (I have no idea what that is)

Fix: Fixed I had to edit the build config in platform.ini

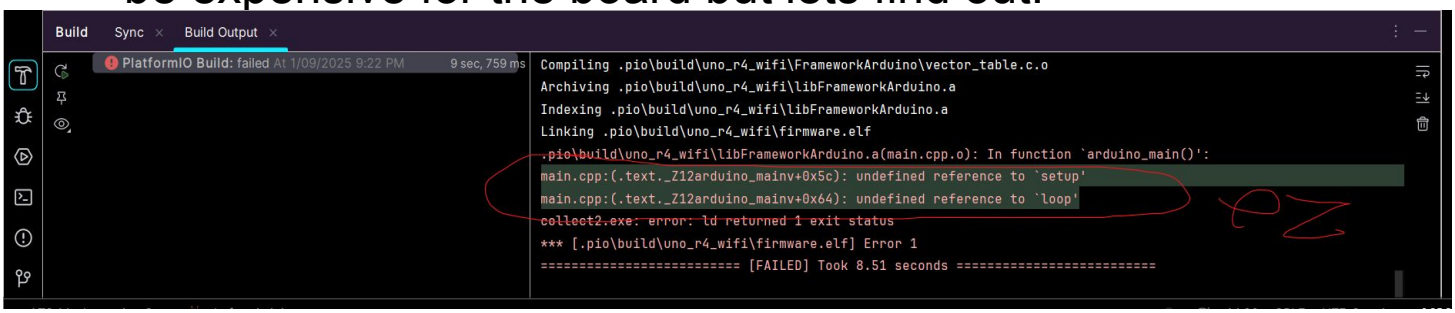




NEW BUG: after that bug solution another error occurred -_-



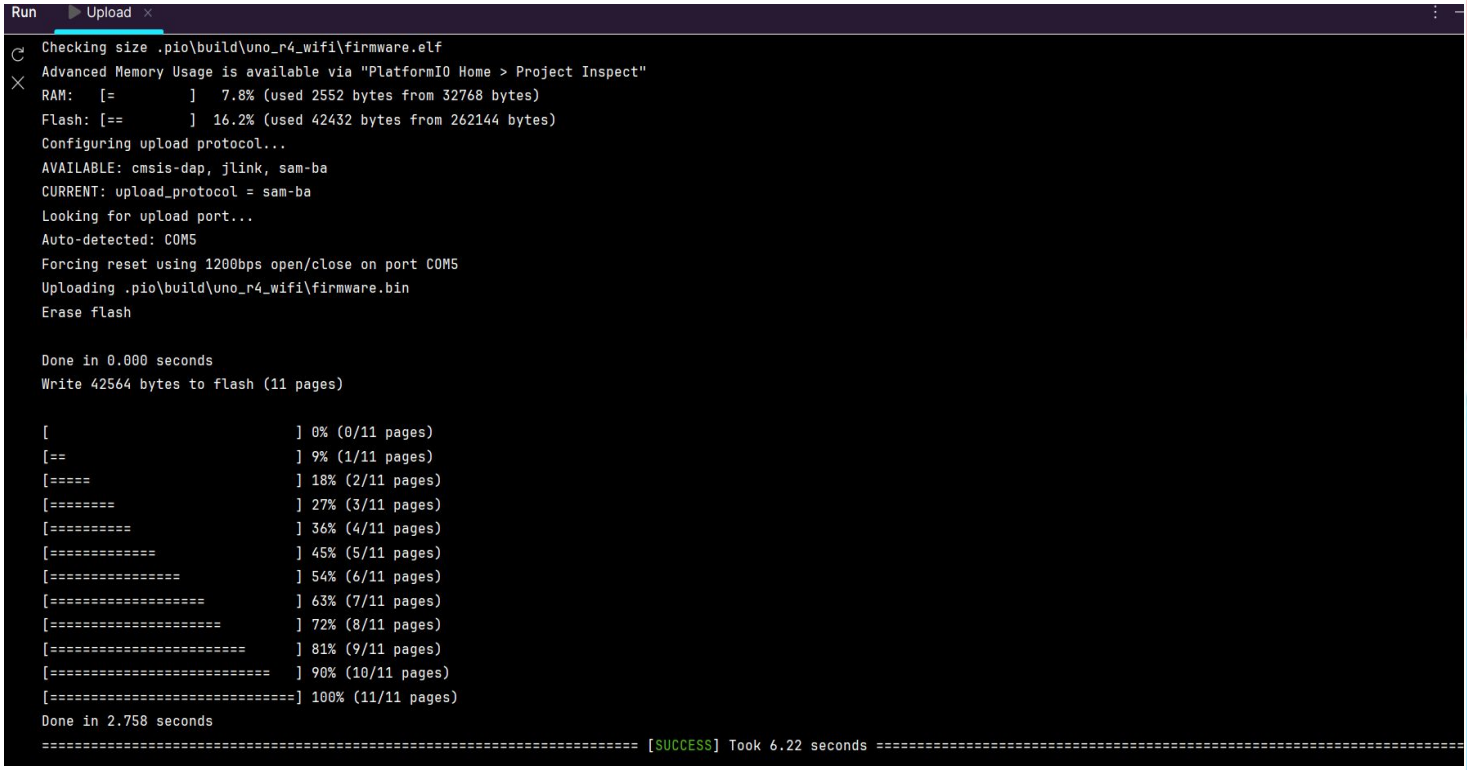
- This was my fault I forgot to define loop and setup (I thought I did) turns out I did it in the wrong place to I had to make a new instance to call class methods (this might be expensive for the board but lets find out.



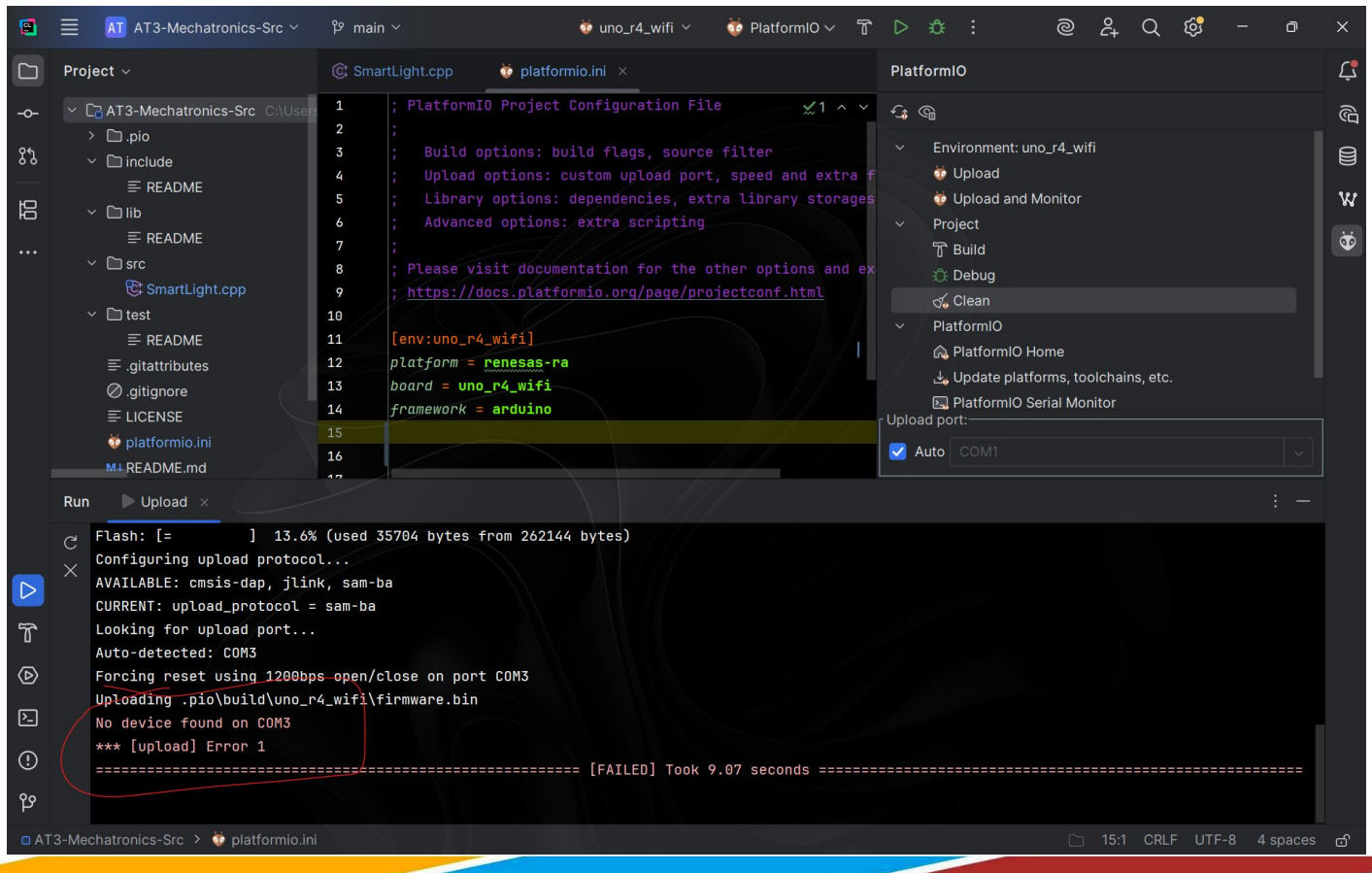
- Solution add this code to bottom:(setup and loop functions missing)

```
195
196 SmartLight::SmartLight SmartLightSystem;
197 //outside of namespace for same method
198 void setup(){SmartLightSystem.setup();};
199 void loop(){SmartLightSystem.loop();};
200
```

- Code build and compiled and runned this time!
  - But will sensors work?
    - still my biggest concern (1 did not 😭)

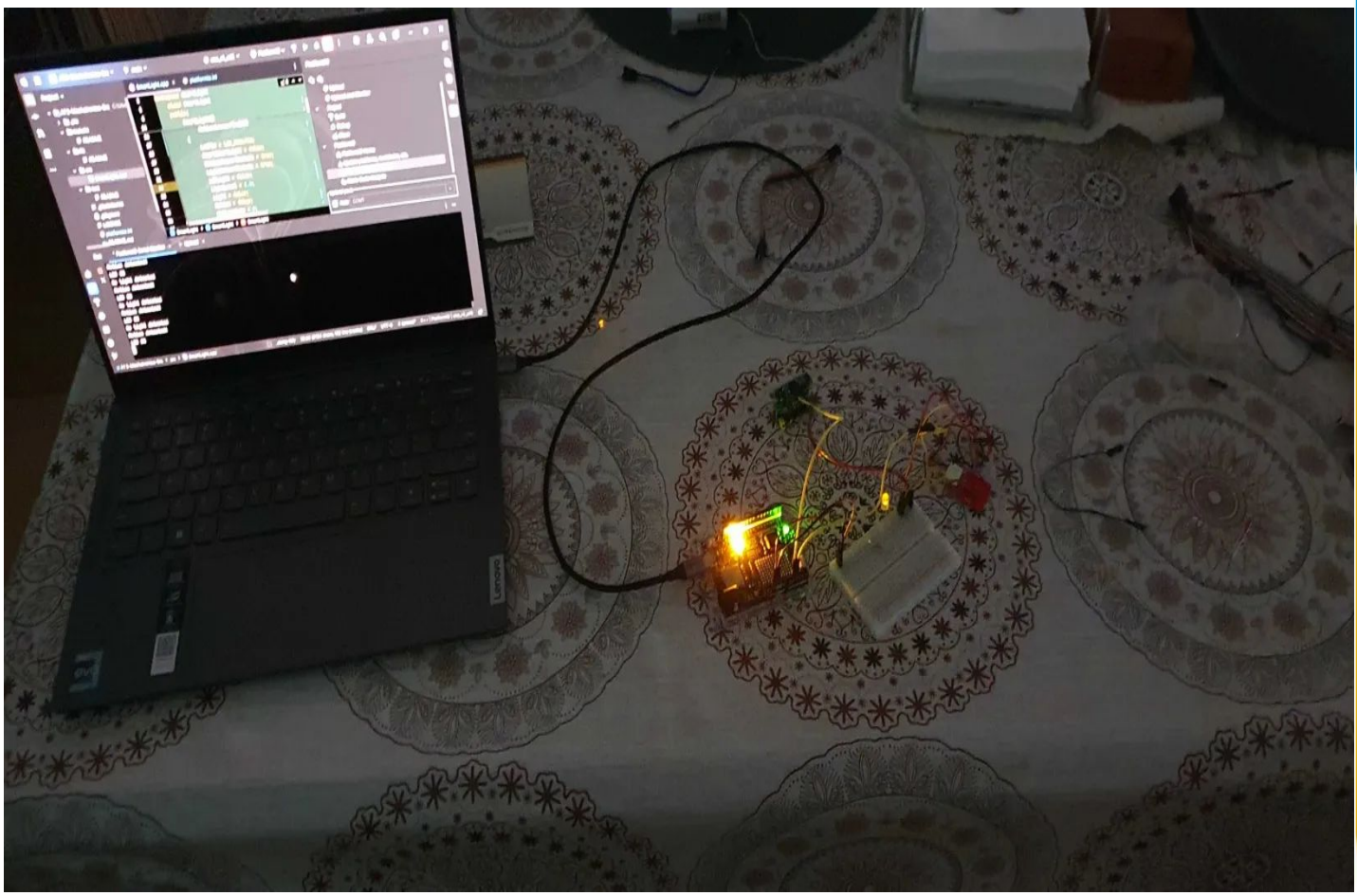


- New error no device detected (code not uploading)
- Solution: used a different usb (it worked)

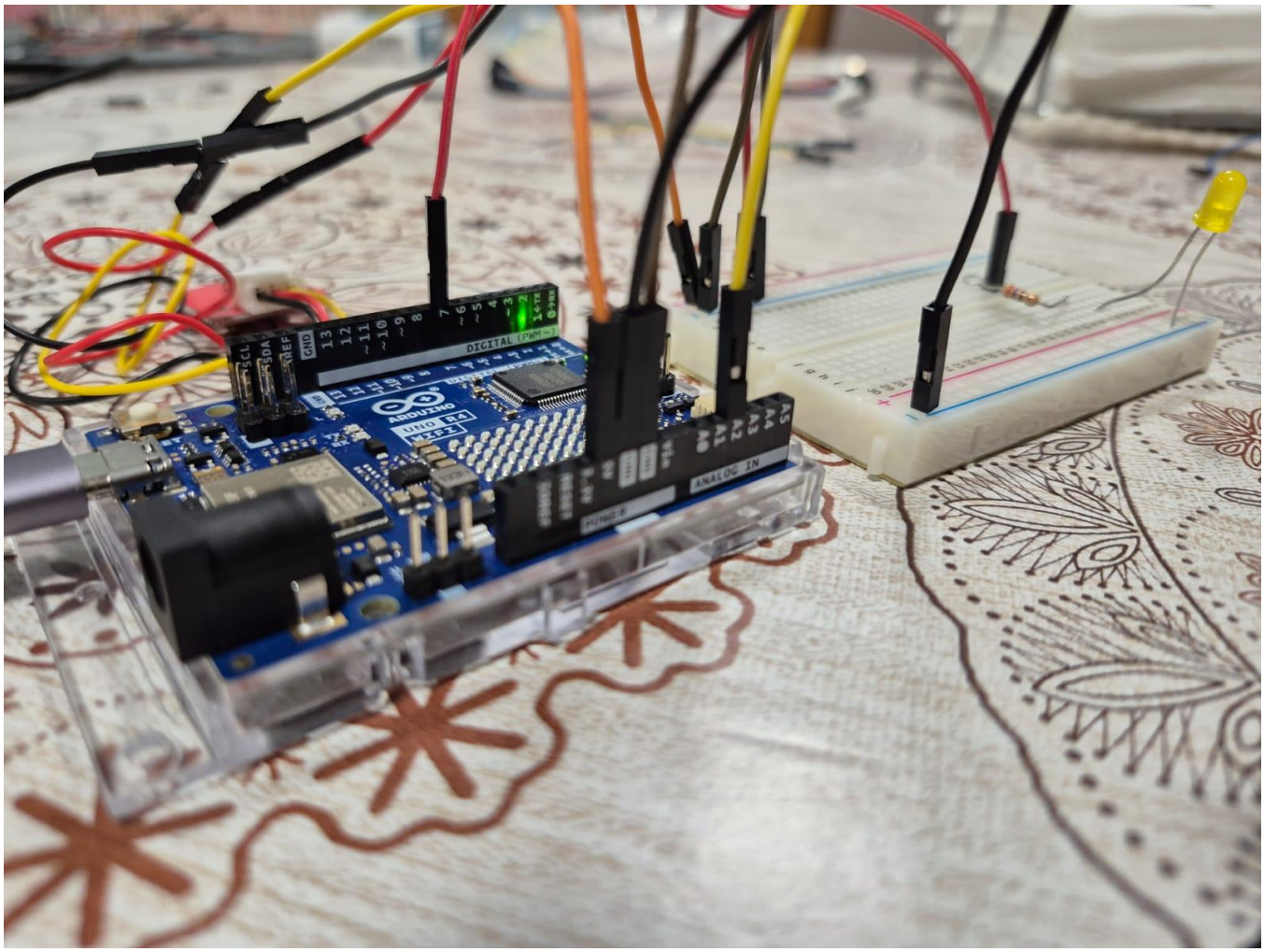


- Another error occurred
  - Type: hardware
  - What happened: the led light turns on even when the programs code is detecting light and condition is set to LED OFF
    - This 100% means that the issue is wiring related as the code is trying to set led light too off but the hardware setup is not correct.

- This was the biggest issue in my project and it took me almost 4 hours to fix.
  - I had to rewire and remove the motion sensor from the project as it was false detecting motion and messing with code condition.
- fix :
  - After countless hours of rewiring and hoping it would it finally worked!
- Incorrect cabling:



- Correct cabling:



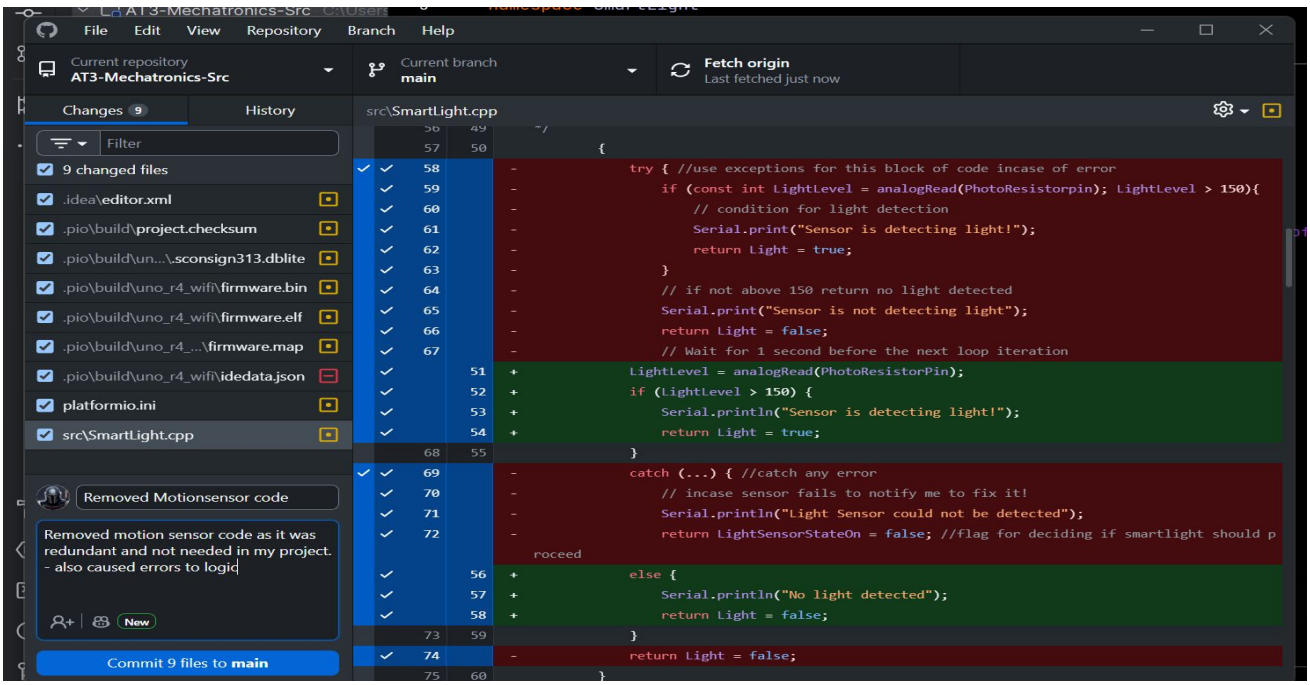
- Had to set the led light on the other side of the board to avoid unwanted current

- Connected 2 cables positive to resistor and negative to ground to the arduino pin D7 pin which allowed the board to control when to send current to the part and light up the led
    - Also switched resistor to allow for a brighter led light.
    - **Removed motion sensor from board and wiped it's code** (still kept the design process of it to show evidence of thought).

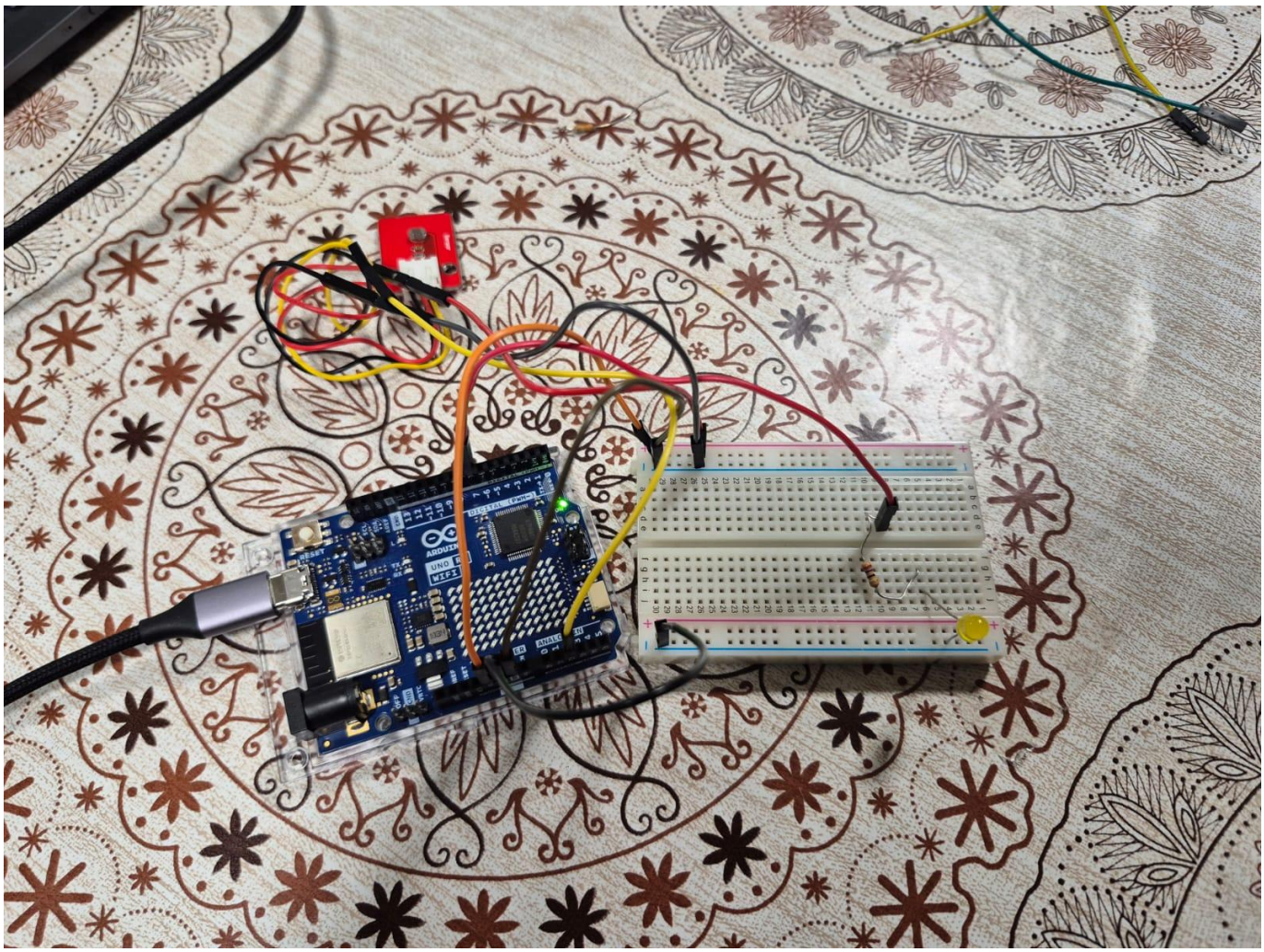
- I removed the motion sensor part from my project was the use of the motion sensor was redundant for my project due to the light sensor and it was also false detecting motion which made me think the sensor was faulty.

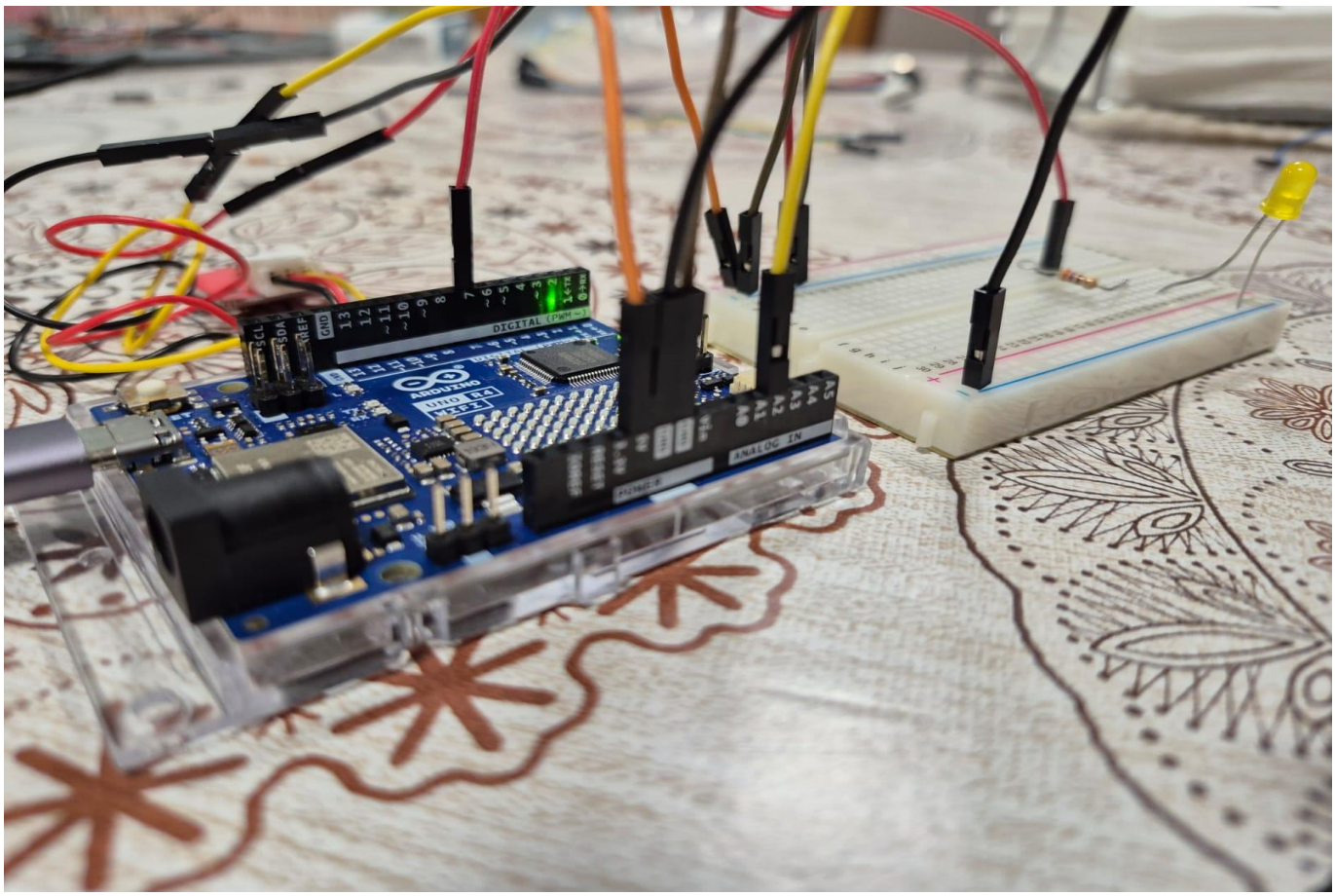
- Due to limitations of time it was a good idea to remove it so I could focus on getting the project wiring to work.

- It took almost 4 hours from 6-10pm to complete the wiring due to the amount of minor errors of wiring (the led light would either turn on even if it was detecting light or not turn on at all.
    - Turns out the reason was because of my wiring and the pin assignment was incorrect (the code could not find what to set to low.
        - I reassigned the pin to the value **D7** and it was working however the brightness was low
        - The solution was to use a different resistor I also tried different leds and found the best one to use for clear view.

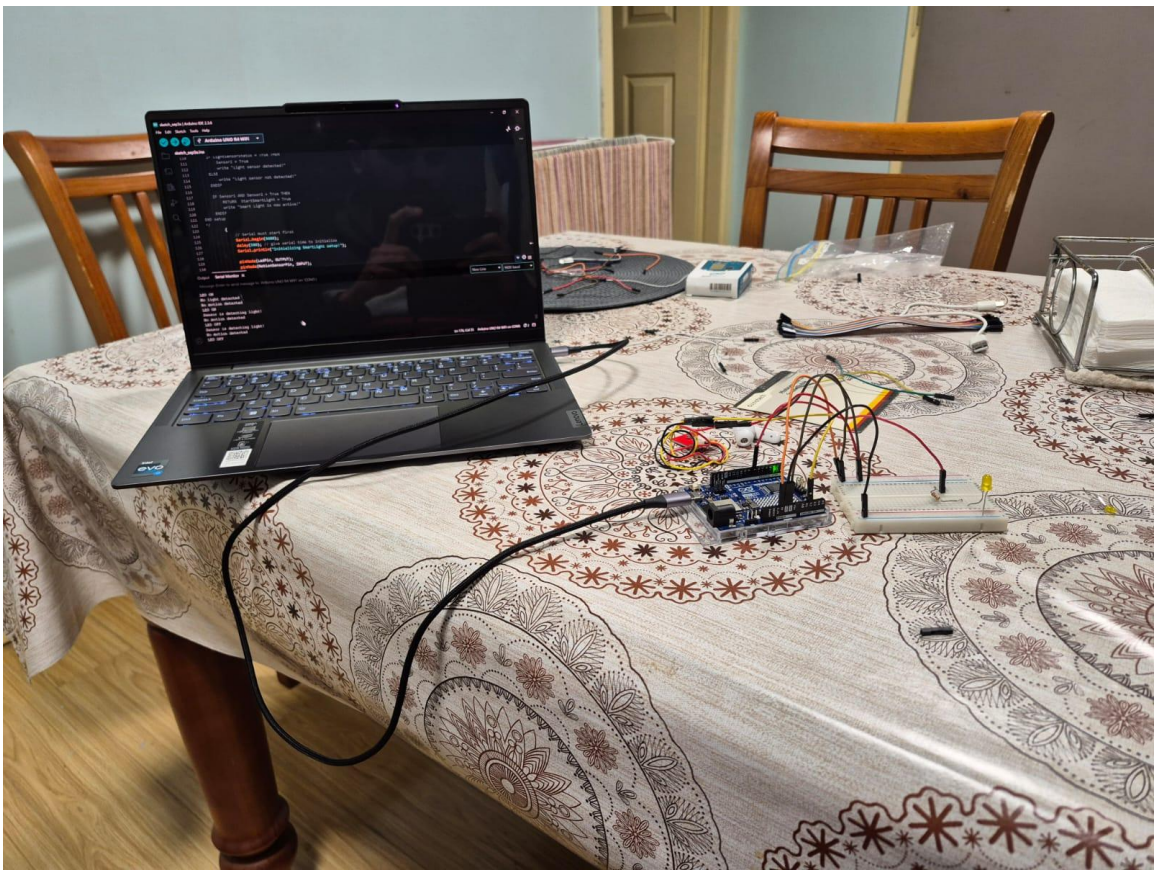


- The final board:

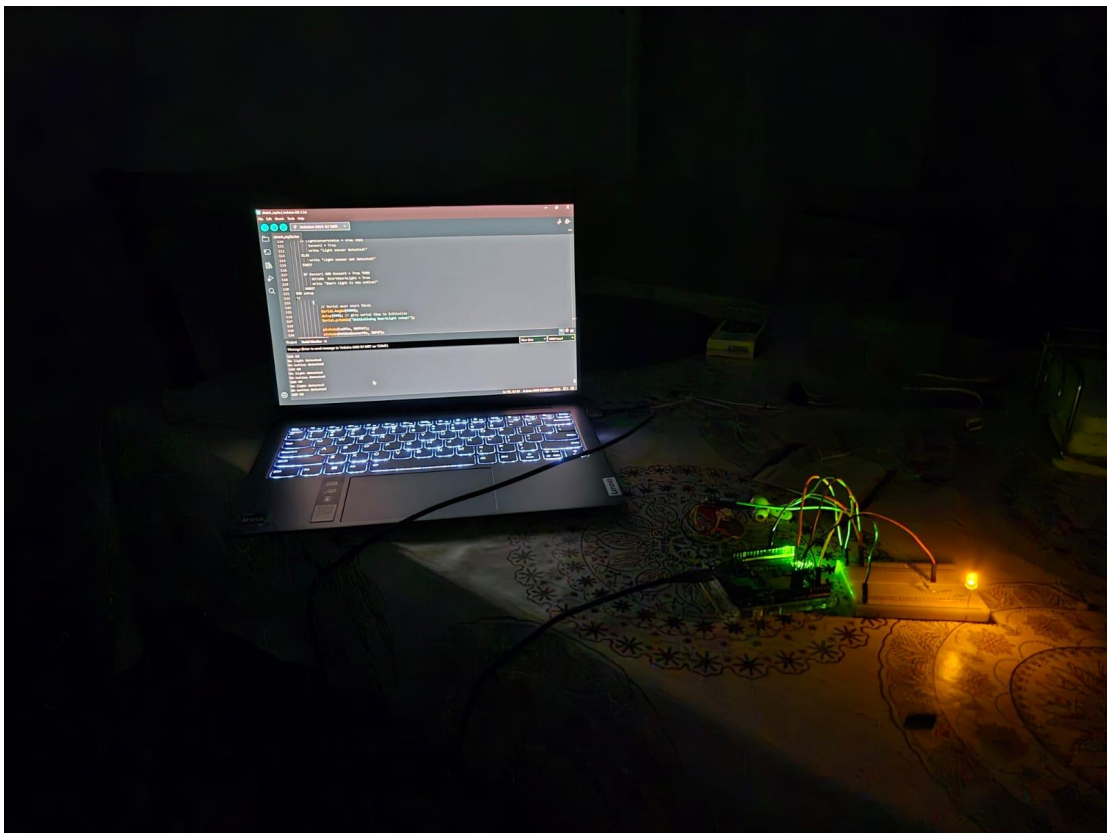




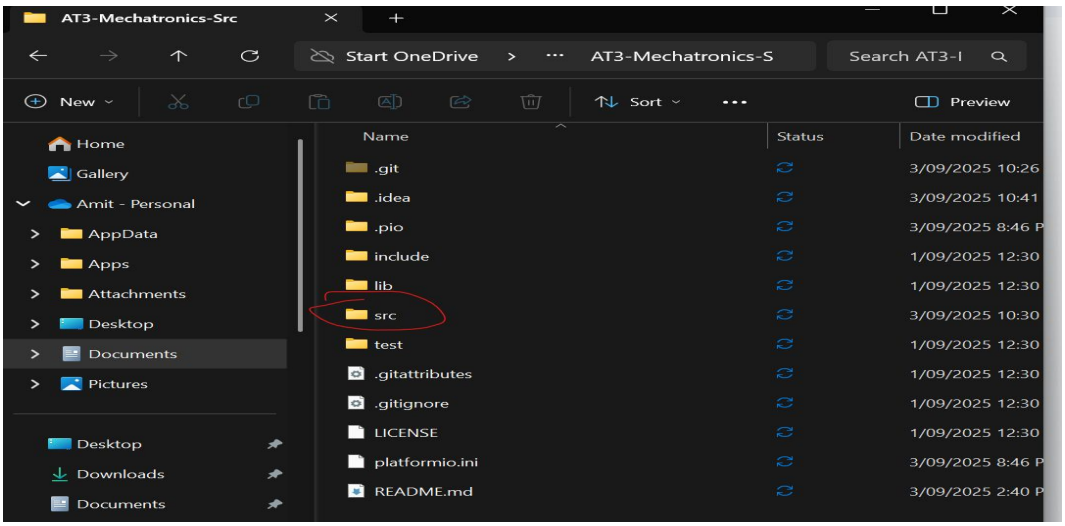
- **The final result:**
  - **Day (light detected):**



- **Night no light detected:**



- **FInal changes:**
  - Converting into an ino to make sure it works on the arduino ide.

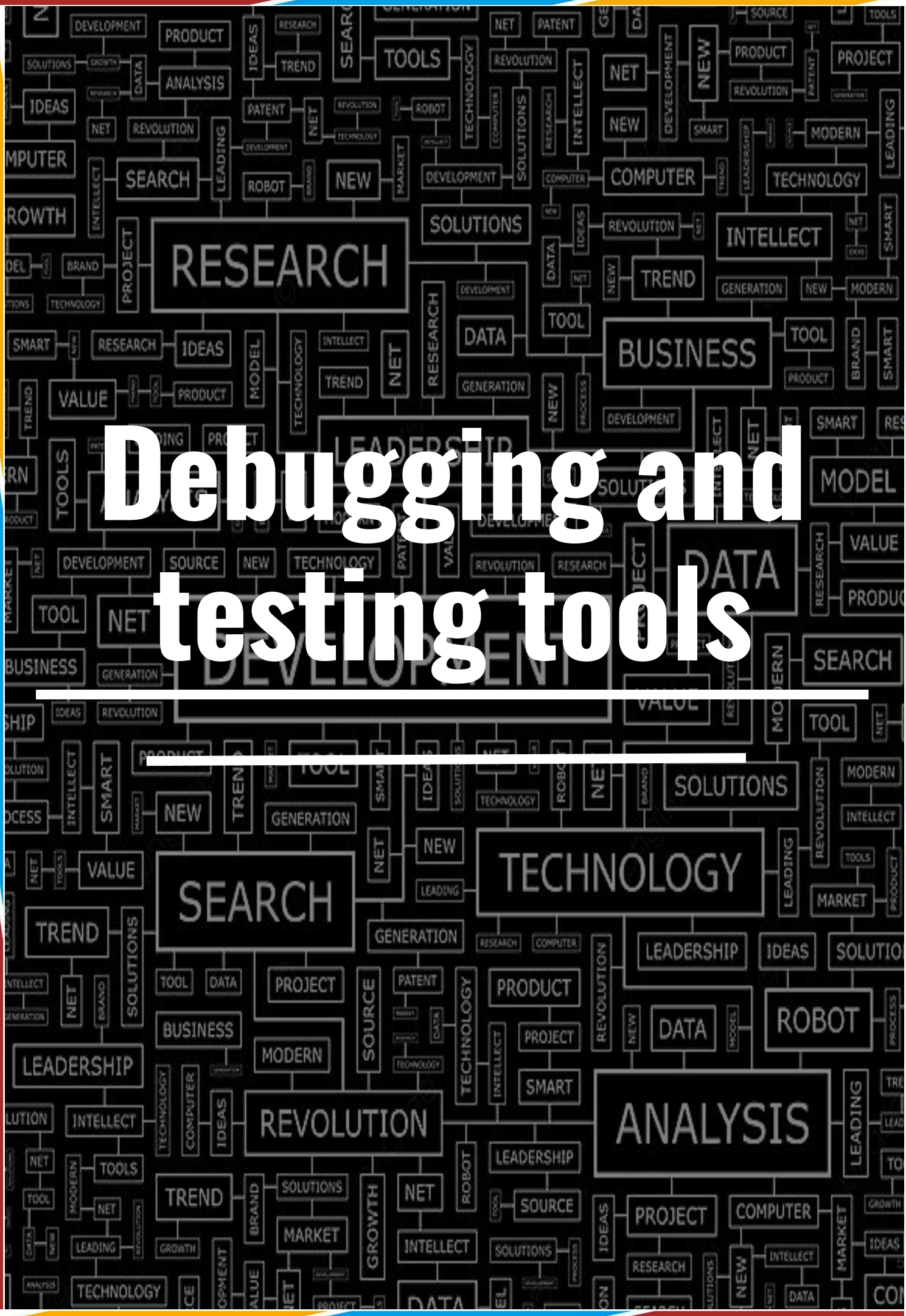


- Platformio (cpp file)
- arduino ide (ino file)
- Removed exceptions from code to keep it simple (board might also explode if I did keep exceptions)


Discussion of data structures

Data dictionary

Variable	Data type	Format for display	Size in bytes	Size for display	Description	Example	Validation
LEDLight	Boolean	X	1 (typical)	1	Output state of LED; ON when both sensors detect conditions.	Y	Y or N
LightLevel	Double (ADC reading)	NNN..N	Platform-dependent (4–8)	Up to 5 digits	Raw light intensity from photoresistor via `analogRead()`. Typically 0–1023.	768	Integer 0–1023
Light	Boolean	X	1 (typical)	1	Derived flag from `LightSensor()`: True if LightLevel > 150.	Y	Y or N
Motion	Boolean	X	1 (typical)	1	Derived flag from `MotionSensor()`: True when PIR detects movement.	N	Y or N
MotionSensorStateOn	Boolean	X	1 (typical)	1	Health flag set False if Motion sensor read throws error.	Y	Y or N
LightSensorStateOn	Boolean	X	1 (typical)	1	Health flag set False if Light sensor read throws error.	Y	Y or N
StartSmartLight	Boolean	X	1 (typical)	1	System enable flag set True when both sensors are healthy in setup().	Y	Y or N
LedPin	Integer (pin)	NN	2–4	2	Arduino pin number driving the LED (initialised to A1).	A1	Valid digital/analog pin for board
PhotoResistorpin	Const Integer (pin)	NN	2–4	2	Analog pin connected to photoresistor (A1).	A1	Valid analog-capable pin
MotionSensorPin	Const Integer (pin)	NN	2–4	2	Pin connected to PIR motion sensor (A1 in stub; should be distinct in wiring).	A1	Valid digital pin; avoid sharing pins
Motionstate	Integer	N	2	1	Last `digitalRead()` value from PIR: HIGH=1, LOW=0.	1	0 or 1


Debugging and
testing tools

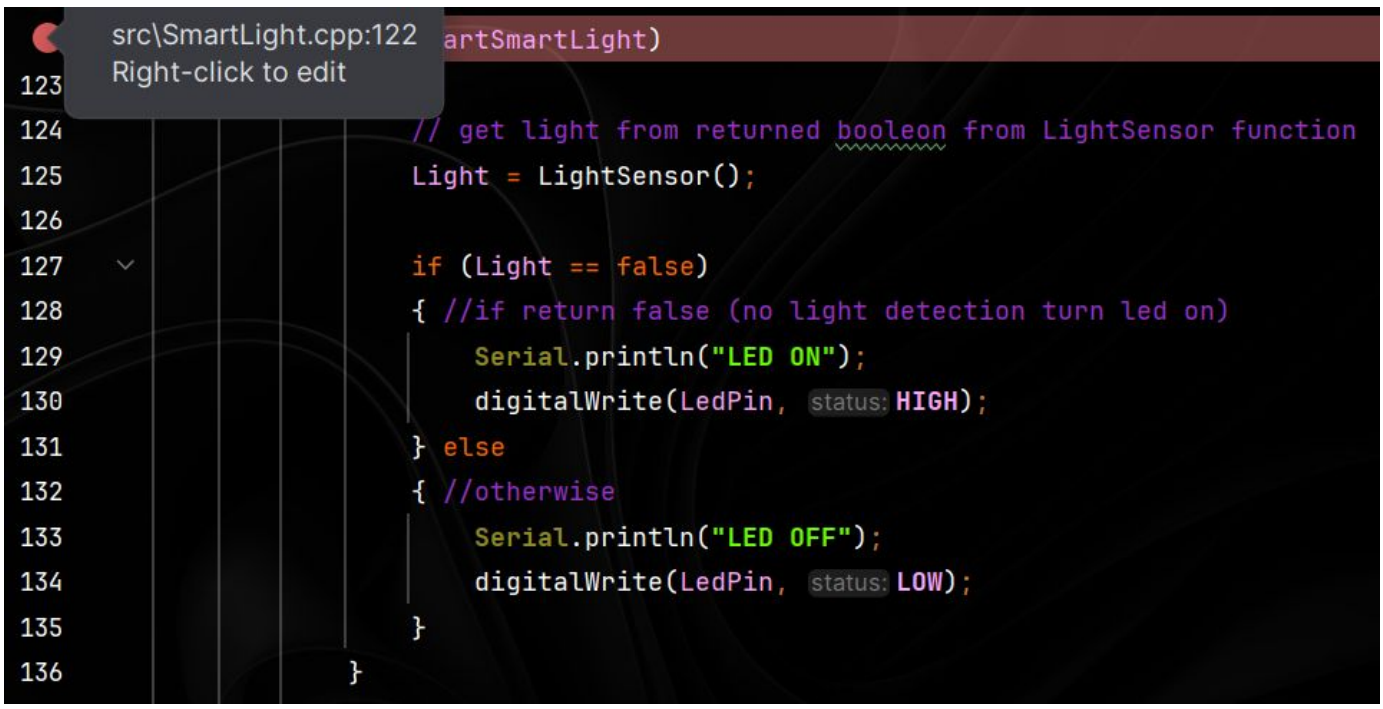
Break points

Breakpoints: Breakpoints are special markers that suspend program execution at a specific point. This lets you examine the program state and behavior. Breakpoints can be simple, for example, suspending the program on reaching some line of code, or involve more complex logic, such as checking against additional conditions, writing to a log, and so on).

Use: used to examine the behavior of the code in my project.

Types:

- *Line breakpoints*: suspend the program upon reaching the line of code where the breakpoint was set. This type of breakpoints can be set on any executable line of code.
- *Exception breakpoints*: suspend the program when the specified exception is thrown. They apply globally to the exception condition and do not require a particular source code reference.
- *Symbolic breakpoints*: suspend the program when a specific function or method are executed.

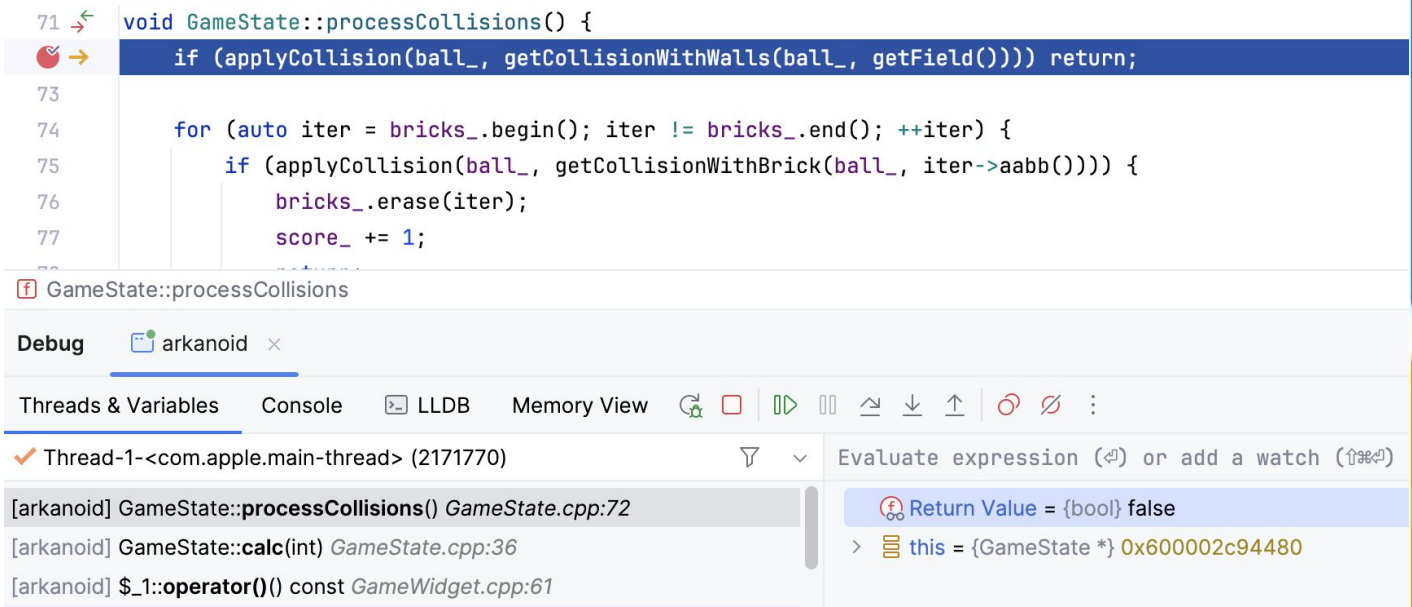


single line stepping

Single line stepping: Single line stepping can be done when you have stopped at a breakpoint where you can execute the code line by line like it works normally, however it freezes the code at every step so you can examine and manipulate it to test what you want to test. As you go through each line, any assigned variables are recorded, any changes in variables are updated and any inputs that are taken will be added too.

In Clion there a different types of line stepping:

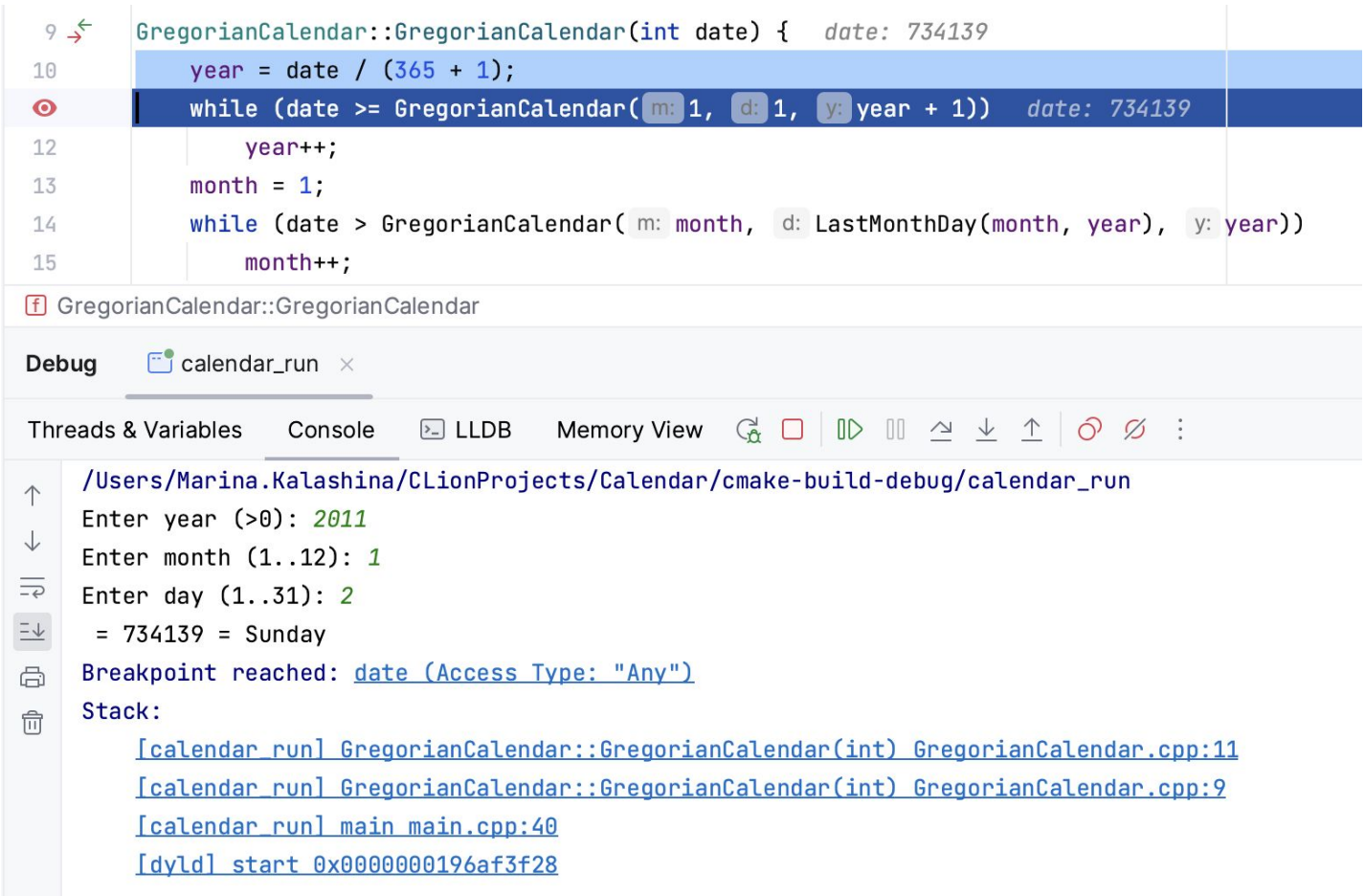
See documentation



watches

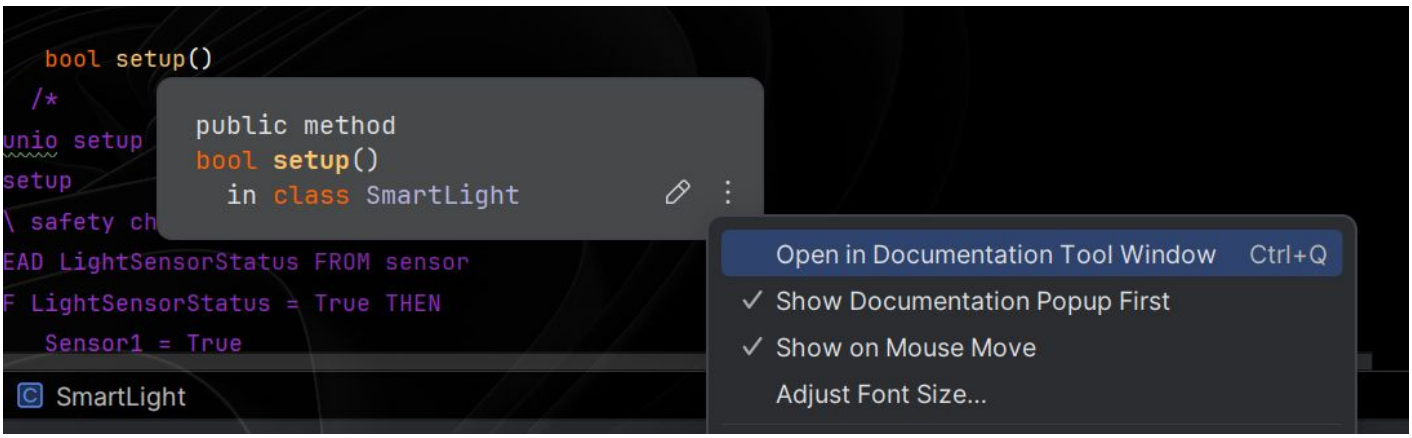
Watches: When you are debugging, you determine which variables are impacting your code and which one's you need to view as the error is coming from them. Once you have identified which variables you want to watch, you can add them into your watch list which will always keep that variable on your screen, making it easy to see it change as you go through your code.

Watchpoints in clion: [Watchpoints | CLion Documentation]



interfaces between functions

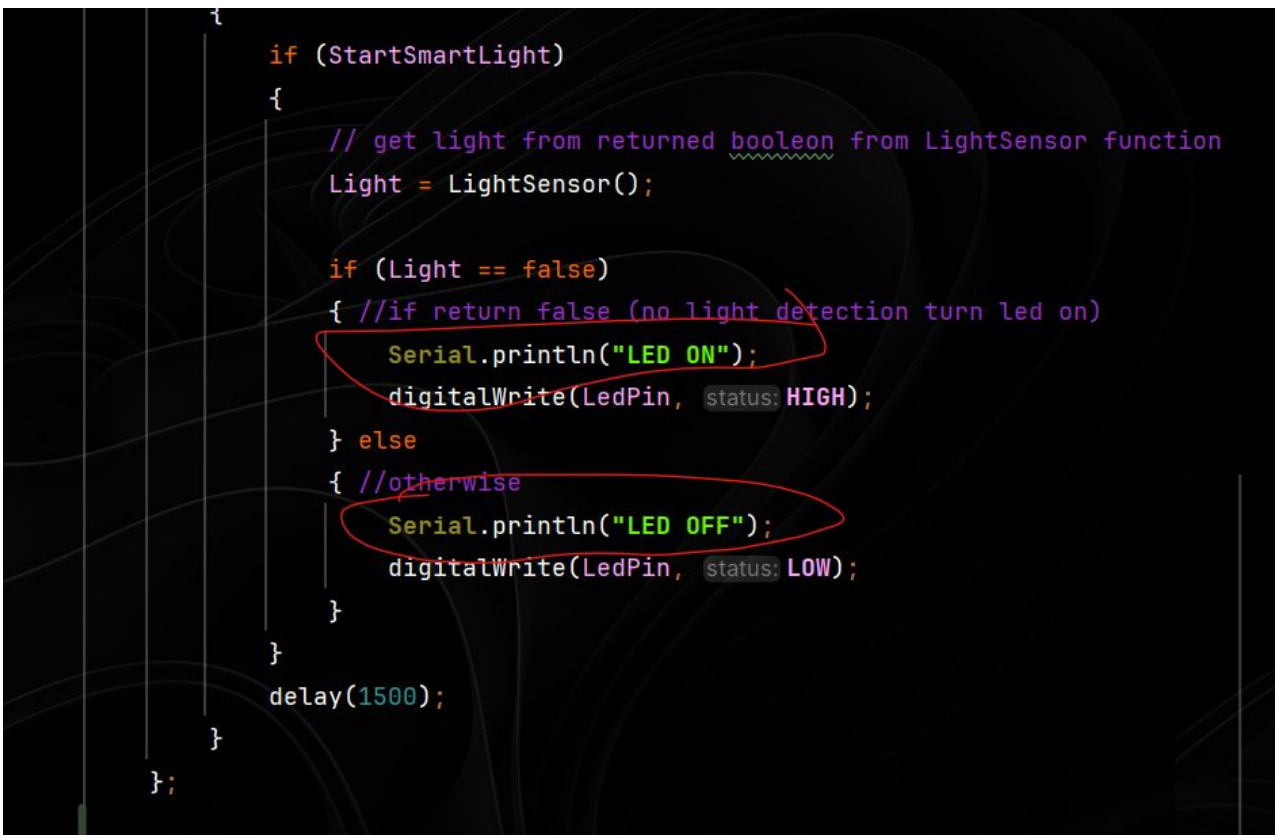
Interface between functions: This refers to ensuring that when functions are being dealt with, the correct amount of parameters are passed into it, the correct type of parameters are given because the function may not be able to use a different data type to what it expects and if the function is returning a value then when it is called, there should be some variable assigned to it or it should be used in such a way that the value returned is used. These are the common errors in functions which can be debugged by ensuring that all the parameters and returned values are used correctly.

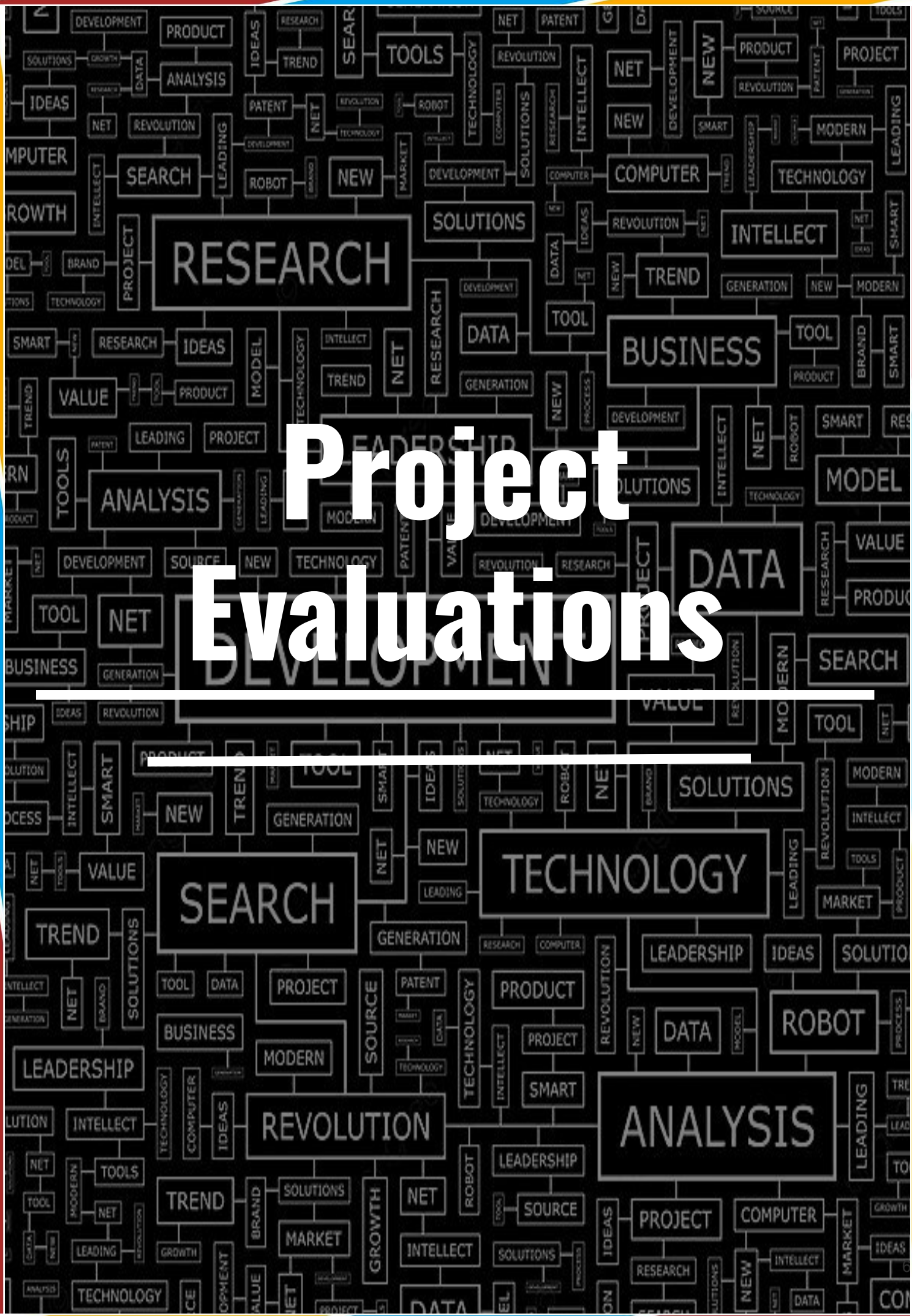


Debug output statements

When debugging, it is effective to add a print statement after or before a line to see whether your code is functional up to that line. If you are unsure of what is happening to a variable at a certain stage of the code then you can print that variable and run the code. By doing this, you can debug the errors in your code simply by displaying the error via an output statement.

Also by adding the try code block you can easily catch any fix errors without the whole app crashing (not used for final because clion exceptions are terrible) instead it will display an error message such could not find reference. I used various debug statements using the Serial.println method to determine what the sensor are seeing such as if they are on or trying to turn the LED light on.




Project
Evaluations

Overall, this project was a successful demonstration of how mechatronics and object-oriented programming can be applied to solve a real-world problem. My goal was to create a smart lighting system that automatically responds to environmental conditions, and despite challenges, I was able to produce a working prototype.

Strengths of the project:

- The system correctly detected light levels and controlled the LED based on conditions.
- Object-oriented programming principles were applied, making the code more organised and modular.
- Debugging strategies such as serial output, breakpoints, and step-through testing improved the reliability of the final design.
- The project aligned well with the original intent of saving energy and encouraging eco-friendly automation.

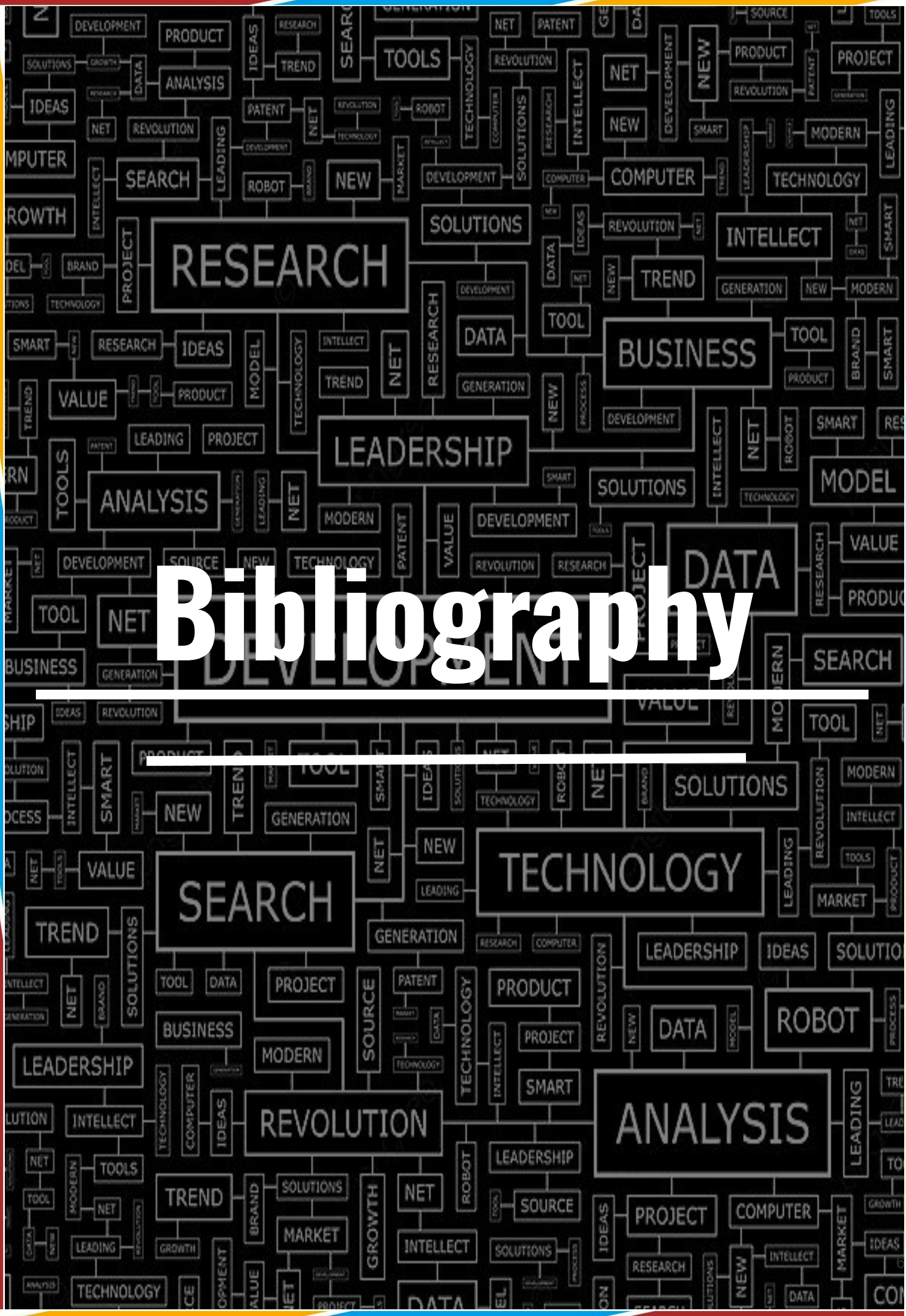
Limitations of the project:

- The prototype only simulated lighting with a small LED rather than a full-scale lamp.
- The motion sensor was removed due to hardware errors, reducing the system's functionality.
- Breadboard wiring was unreliable and caused time-consuming troubleshooting.
- Thresholds for light detection were hard-coded instead of adjustable through a user interface.

Reflection:

This project allowed me to develop skills in Arduino programming, C++ OOP, and debugging hardware/software interactions. I learned that planning the algorithm and structure chart before coding reduced errors later. I also discovered the importance of careful wiring and component testing, as hardware issues often took longer to resolve than coding problems.

Overall, my smart light prototype demonstrated an energy-efficient lighting solution. While there are areas for improvement, the project was a valuable learning experience that successfully addressed the chosen real-world problem and met the assessment requirements, .



Bibliography

YouTube (2025) *Smart light automation with Arduino*. Available at: https://www.youtube.com/watch?v=W6mixXsn-Vc (Accessed: 12 August 2025).

Rodlo, M. (2025) *Arduino UNO R4 examples*. GitHub. Available at: https://github.com/matiasrodlo/arduino-uno-r4 (Accessed: 23 August 2025).

Airstriker123 (2025) *Structure chart symbols*. GitHub. Available at: https://github.com/Airstriker123/Structure-chart-symbols (Accessed: 25 August 2025).

Airstriker123 (2025) *AT3 Mechatronics source code*. GitHub. Available at: https://github.com/Airstriker123/AT3-Mechatronics-Src (Accessed: 29 August 2025).

Arduino (2025) *Arduino documentation*. Available at: https://docs.arduino.cc/ (Accessed: 4 September 2025).

Cplusplus Reference (2025) *C++ reference documentation*. Available at: https://en.cppreference.com/w/ (Accessed: 6 September 2025).

Informer Software (2025) *LARP 3.0 download page*. Available at: https://larp.software.informer.com/3.0/ (Accessed: 9 September 2025).

diagrams.net (2025) *Online diagramming tool*. Available at: https://app.diagrams.net/ (Accessed: 11 September 2025).

PlatformIO (2025) *PlatformIO project examples*. GitHub. Available at: https://github.com/platformio/platformio-examples (Accessed: 14 September 2025).

JetBrains (2025) *CLion IDE*. Available at: https://www.jetbrains.com/clion/ (Accessed: 17 September 2025).

MikeTeachesTech (2025) *HSC Software Engineering resources*. GitHub. Available at: https://github.com/miketeachestech/hsc-software-engineering-resources (Accessed: 19 September 2025).

PlatformIO (2025) *PlatformIO official site*. Available at: https://platformio.org/ (Accessed: 21 September 2025).

JetBrains (2025) *Debugging with CLion – stepping through code*. Available at: https://www.jetbrains.com/help/clion/stepping-through-the-program.html#step-into (Accessed: 23 September 2025).

GeeksforGeeks (2025) *Arduino coding basics*. Available at: https://www.geeksforgeeks.org/electronics-engineering/arduino-coding-basics/ (Accessed: 26 September 2025).

Energy Theory (2025) *Ways to save electricity*. Available at: https://energytheory.com/ways-to-save-electricity/ (Accessed: 29 September 2025).