runlinc Intermediate Project 7: Displaying Light Sensor (E32W Version)

Contents

Introduction

Problem

Light is an important parameter affecting the growth of crops. We want to keep track on the current brightness level of crops. And We want to observe the brightness level over a period to monitor the effect of light on crops.

Background

Light is an essential factor in maintaining plants. The rate of growth and length of time a plant remains active is dependent on the amount of light it receives. Photosynthesis is a topic that is done to death in our science classrooms. Photosynthesis, the process by which green plants and certain other organisms transform light energy into chemical energy. During photosynthesis in green plants, light energy is captured and used to convert water, carbon dioxide, and minerals into oxygen and energy-rich organic compounds. The chemical equation for this process is: $6CO_2 + 12H_2O + light \rightarrow C_6H_{12}O_6 + 6O_2 + 6H_2O$.

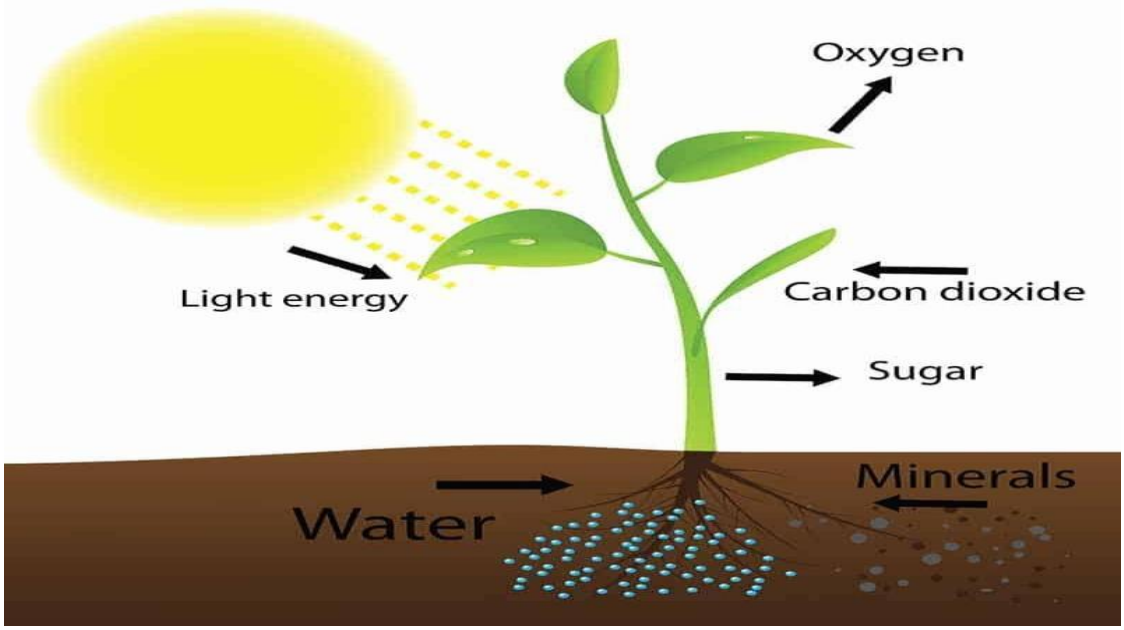

Figure 1: Photosynthesis process

The relationship between photosynthesis and light intensity is illustrated in figure 2.

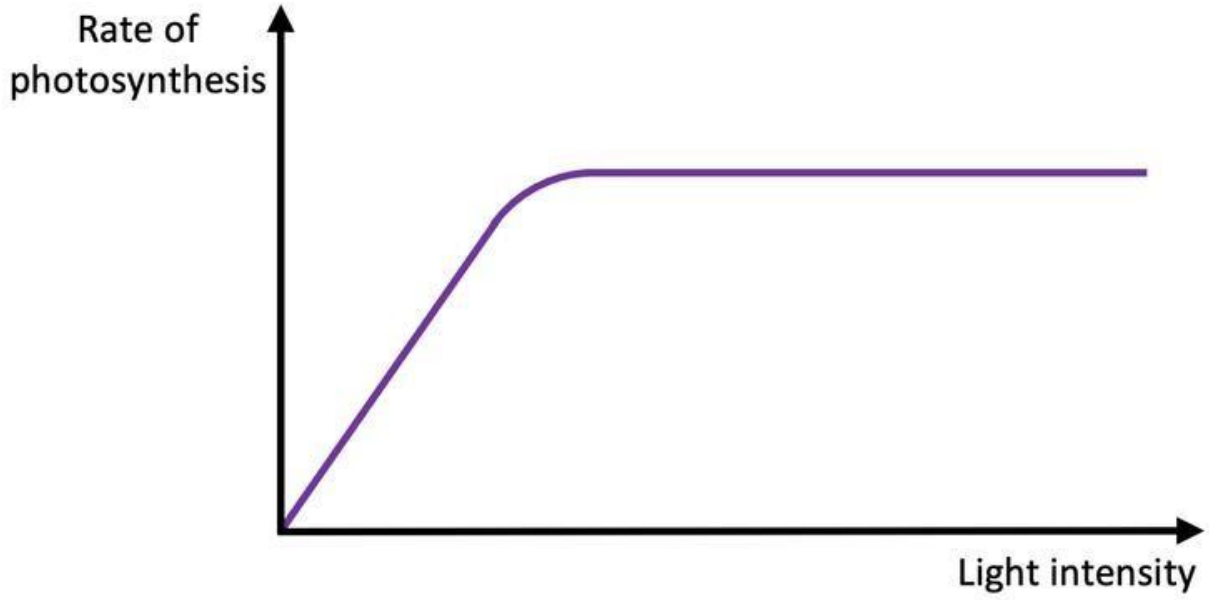

Figure 2: Relationship between photosynthesis and light intensity

So by level of light you probably mean light intensity which is something that can be measured. Light intensity is usually defined as the energy hitting an area over some period. So, in the case of a plant, a higher light intensity means more packets of light called

"photons" are hitting the leaves. As you rise from low light intensity to higher light intensity, the rate of photosynthesis will increase because there is more light available to drive the reactions of photosynthesis. However, once the light intensity gets high enough, the rate won't increase anymore because there will other factors that are limiting the rate of photosynthesis. A limiting factor could be the amount of chlorophyll molecules that are absorbing the light. At a very high intensity of light, the rate of photosynthesis would drop quickly as the light starts to damage the plant. Thus it is necessary to monitor the light intensity of the plants to make the plants grow in a suitable light environment.

Ideas

How can we measure the brightness level? How can the microchip and runlinc measure the real-time light intensity. What do we have that could be used to indicate for the farmer what the brightness level the plants is at for a period?

Plan

We have Light Dependant Resistor (LDR) in our kits which we can use to check the light intensity of a possible fire, and a web input to change the light intensity risk.

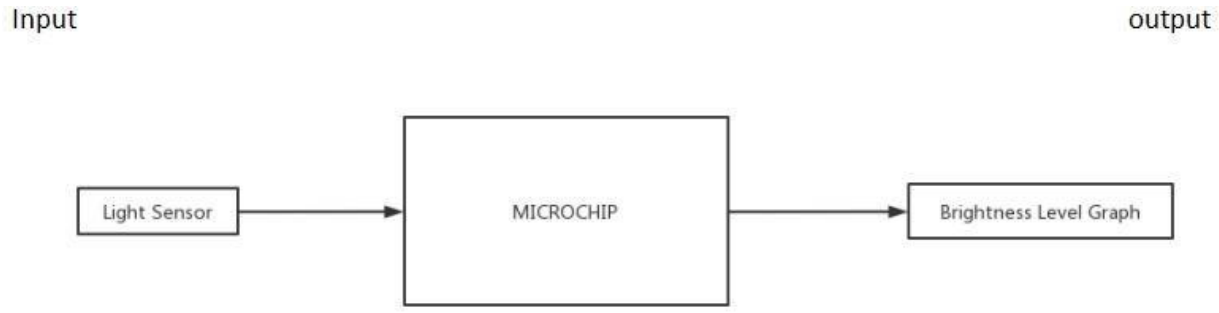

Figure 3: Block diagram of Microchip outputs

runlinc Background

runlinc is a web page inside a Wi-Fi chip. The programming is done inside the browsers compare to programming inside a chip. The runlinc web page inside the Wi-Fi chip will command the microchips to do sensing, control, data logging Internet of Things (IoT). It can predict and command.

Part A: Design the Circuit on runlinc

Note: Refer to runlinc Wi-Fi Setup Guide document to connect to runlinc

Use the left side of the runlinc web page to construct an input/output (I/O).

For port D33 name it LightSensor and set it as ANALOG_IN.

In our circuit design, we will be using the light sensor. We happen to have this in our kits, so these can be used on our circuit design, as per the plan.

Figure 4: I/O configurations connections

Part B: Build the Circuit

Use the STEMSEL E32W board to connect the hardware. For this project we are using both the left and right I/O ports, with **negative port (-ve)** on the outer side, **positive port (+ve)** on the middle and **signal port (s)** on the inner side (as shown below).

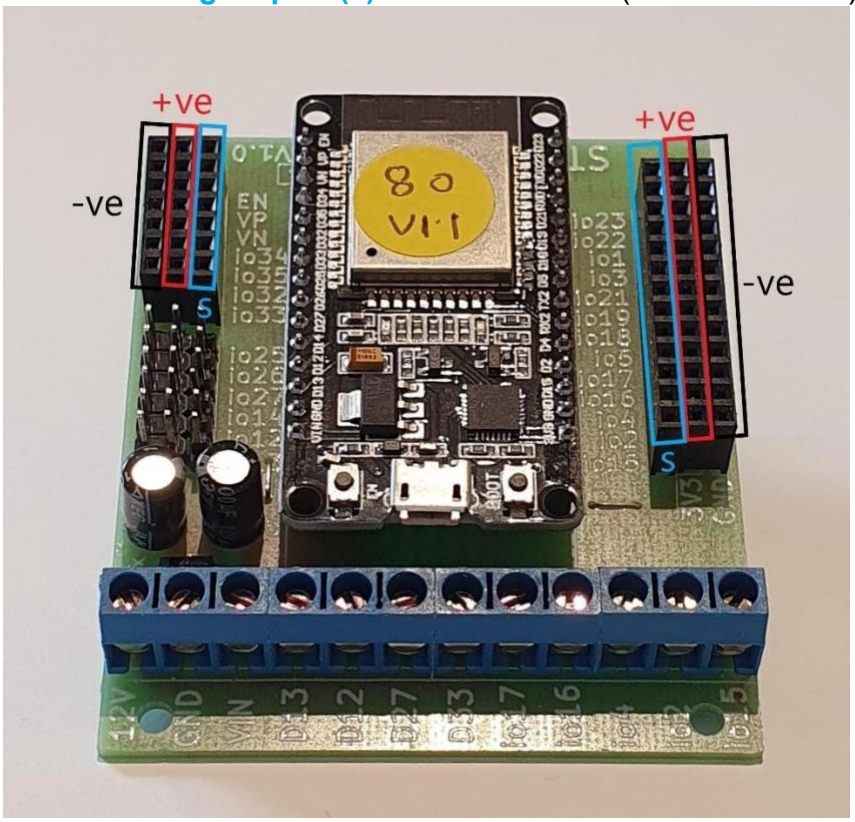

Figure 5: Negative, Positive and Signal port on the E32W board

There is one I/O part we are using for this project, a Light Dependant Resistor (LDR) module (KY-018), their respective pins are shown in the figure below.

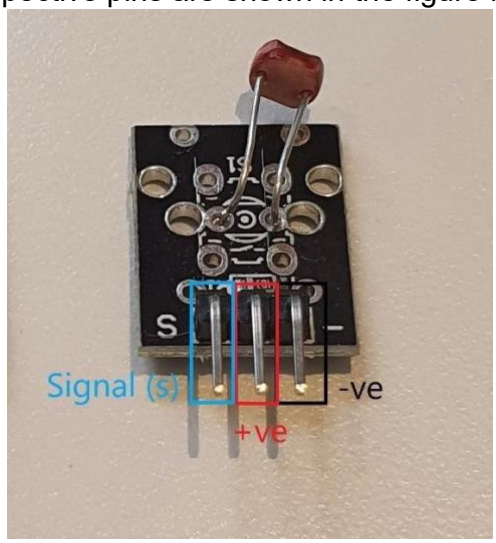

Figure 6: I/O part with negative, positive and signal pins indicated

Wiring instructions

a.) Plug in the Light Dependant Resistor (LDR) to io33 on the E32W board.

b.) Make sure the (-ve) pin are on the GND (outer) side of the I/O ports.

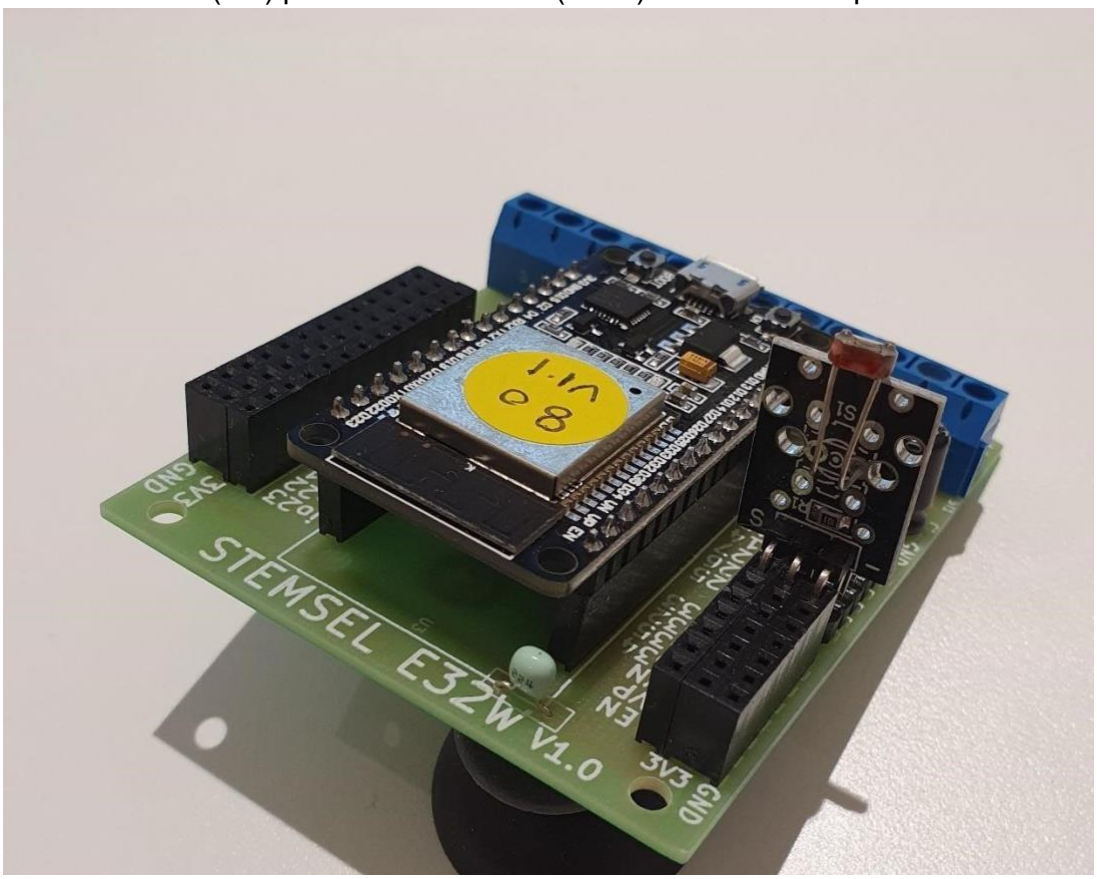

Figure 7: Circuit board connection with I/O part (side view)

Figure 8: Circuit board connection with I/O part (top view)

Part C: Program the Circuit

CSS:

Under CSS section, we will be creating a class that changes the font size to 25 pixels

```
div.a{ font-size: 25px;
}
```

HTML:

Under HTML, we will be displaying the title of the webpage, the current brightness level and the light sensor graph. The font size of the current brightness level will be 25 pixels.

Note: "//" represents comments for each line of code

1. We will first align all contents to the centre, at the same time we'll add a header.

```
<div style = "text-align:center">
    <h1> <font id = "Title" color = "darkblue"> Light Sensor Graph </font> </h1>
```

2. After the header element, we will start to design the Light Sensor graph for displaying the brightness level detected from light sensor. Firstly, we will create the x axis and y axis respectively named "Time" and "Brightness Level". Then set the scale of y axis("Brightness Level") with 0, 50, 100, 150 and 250.

```
<div class = "a">
<td> <font id = "Brightness"> </font> </td>
</div>
<br>
<svg height = "350" width = "600">
    <polyline id = "brightnessGraph" style = "fill: none; stroke: darkblue; strokewidth:3" />
    <text x = "95" y = "315" fill = "black">0</text>
    <text x = "70" y = "50" fill = "black">250</text>
    <text x = "70" y = "100" fill = "black">200</text>
    <text x = "70" y = "150" fill = "black">150</text>
    <text x = "70" y = "200" fill = "black">100</text>
    <text x = "70" y = "250" fill = "black">50</text>
```

```
    <text x = "320" y = "340" fill = "darkblue">Time</text> //x-axis title
    <text x = "0" y = "150" fill = "darkblue" id = "yTitleOne">Brightness</text>//y-axis
title
    <text x = "0" y = "170" fill = "darkblue" id = "yTitleTwo">Level</text>//y-axis title
    <line x1 = "100" y1 = "300" x2 = "600" y2 = "300" stroke = "black"/> //x-axis line
    <line x1 = "100" y1 = "0" x2 = "100" y2 = "300" stroke = "black"/> //y-axis line
</svg>
</div>
```

This will set up our webpage to receive the Information from the LightSensor.

JavaScript:

1. We will start with establishing the following global variables that will be used throughout the program and are not limited within functions. As JavaScript is a single thread programming language, where the instructions are followed step by step.

```
    var Height = 300;
    var Width = 500;
    var StartingWidth = 100;
    var widthPerSample = 5;
    var maxWidthSample = Width / widthPerSample;
    var i = 0;
    var Light; // Declare Light in the global scope
    var BrightnessValue = [];
    var BrightnessElement;

    var updatedBrightnessPoints;

var BrightnessPoints = [];
```

2. We will then write a function that collects the brightness level detected by the light sensor and display them on the Light Sensor Graph as a polyline.

```
function updatedGraph() {
    // Recalculate points based on current BrightnessValue
    BrightnessPoints = [];
    let x = 100; // Reset x coordinate
    for (let j = Math.max(0, BrightnessValue.length - maxWidthSample); j <
BrightnessValue.length; j++) {
        let y = Height - BrightnessValue[j];
        BrightnessPoints.push(x + "," + y);
        x += widthPerSample;
    }

    BrightnessElement = document.getElementById('brightnessGraph');
    updatedBrightnessPoints = BrightnessPoints.join(" ");

    BrightnessElement.setAttribute('points', updatedBrightnessPoints);
}

function displayValues() {
    document.getElementById('Brightness').innerHTML = "Current Brightness Level:"
+ Light;
}
```

3. At last, we will write a function to display the highest, current, and lowest brightness level of the light sensor to the webpage.

JavaScript Loop:

In JavaScript Loop, the codes will display the graph with the functions set up in JavaScript in an endless loop.

```
// No need for a while loop, the JavaScript Loop section
already loops!
await mSec(1000);
Light = analogIn(LightSensor);
BrightnessValue.push(Light);
updatedGraph();
displayValues();
```

Final Code:

The final code for **CSS** block:

```
div.a{
font-size: 25px;
}
```

The final code for **HTML** block:

```
<div style = "text-align:center">
<h1><font id = "title" color = "darkblue"> Light Sensor Graph </font></h1>

<div class = "a">
<td><font id = "Brightness"></font></td>
</div>
<br>

<svg height = "350" width = "600">
    <polyline id = "brightnessGraph" style = "fill:none; stroke: darkblue; stroke-width:3" />
    <text x = "95" y = "315" fill = "black">0</text>
    <text x = "70" y = "50" fill = "black">250</text>
    <text x = "70" y = "100" fill = "black">200</text>
    <text x = "70" y = "150" fill = "black">150</text>
    <text x = "70" y = "200" fill = "black">100</text>
    <text x = "70" y = "250" fill = "black">50</text>
    <text x = "320" y = "340" fill = "darkblue">Time</text>
    <text x = "0" y = "150" fill = "darkblue" id = "yTitleOne">Brightness</text>
    <text x = "0" y = "170" fill = "darkblue" id = "yTitleTwo">Level</text>
    <line x1 = "100" y1 = "300" x2 = "600" y2 = "300" stroke = "black"/>
    <line x1 = "100" y1 = "0" x2 = "100" y2 = "300" stroke = "black"/>
</svg>
</div>
```

The final code for **JavaScript** block:

```
var Height = 300;
var Width = 500;
var StartingWidth = 100;
var widthPerSample = 5;
var maxWidthSample = Width / widthPerSample;
var i = 0;
var Light; // Declare Light in the global scope
var BrightnessValue = [];
var BrightnessElement;

var updatedBrightnessPoints;

var BrightnessPoints = [];

function updatedGraph() {
   // Recalculate points based on current
BrightnessValue
   BrightnessPoints = [];
   let x = 100; // Reset x coordinate
   for (let j = Math.max(0, BrightnessValue.length -
maxWidthSample); j < BrightnessValue.length; j++)
{
      let y = Height - BrightnessValue[j];
      BrightnessPoints.push(x + "," + y);
      x += widthPerSample;
   }

   BrightnessElement =
document.getElementById('brightnessGraph');
   updatedBrightnessPoints =
BrightnessPoints.join(" ");

   BrightnessElement.setAttribute('points',
updatedBrightnessPoints);
}
```

```
function displayValues() {

document.getElementById('Brightness').innerHTML
= "Current Brightness Level:" + Light;
}
```

The final code for **JavaScript Loop** block:

```
// No need for a while loop,
the JavaScript Loop section
already loops!
await mSec(1000);
Light =
analogIn(LightSensor);
BrightnessValue.push(Light);
updatedGraph();
displayValues(); }
```

Expected Result

Once you filled the codes in their respective section in the runlinc control page and once everything was ready. When we run the code the webpage will display as shown in Figure 10. However, the program will run continuously. To reset the graph, you need to reload the webpage.

CSS

```
div.a{
      font-size:25px;
}
```

HTML

```
<div style = "text-align:center">
<h1><font id = "title" color = "darkblue"> Light Sensor Graph </font></h1>

<div class = "a">
<td><font id = "Brightness"></font></td>
</div>
<br>

<svg height = "350" width = "600">
        <polyline id = "brightnessGraph" style = "fill:none; stroke:
darkblue; stroke-width:3" />
        <text x = "95" y = "315" fill = "black">0</text>
        <text x = "70" y = "50" fill = "black">250</text>
        <text x = "70" y = "100" fill = "black">200</text>
        <text x = "70" y = "150" fill = "black">150</text>
        <text x = "70" y = "200" fill = "black">100</text>
        <text x = "70" y = "250" fill = "black">50</text>
        <text x = "320" y = "340" fill = "darkblue">Time</text>
        <text x = "0" y = "150" fill = "darkblue" id =
"yTitleOne">Brightness</text>
        <text x = "0" y = "170" fill = "darkblue" id =
"yTitleTwo">Level</text>
        <line x1 = "100" y1 = "300" x2 = "600" y2 = "300" stroke = "black"/>
        <line x1 = "100" y1 = "0" x2 = "100" y2 = "300" stroke = "black"/>
</svg>
</div>
```

JavaScript | Select Macro | select a device | Add Macro

```
var Height = 300;
var Width = 500;
var StartingWidth = 100;
var widthPerSample = 5;
var maxWidthSample = Width / widthPerSample;
var i = 0;
var Light; // Declare Light in the global scope
var BrightnessValue = [];
var BrightnessElement;

var updatedBrightnessPoints;

var BrightnessPoints = [];

function updatedGraph() {
    // Recalculate points based on current BrightnessValue
    BrightnessPoints = [];
    let x = 100; // Reset x coordinate
    for (let j = Math.max(0, BrightnessValue.length - maxWidthSample); j <
BrightnessValue.length; j++) {
        let y = Height - BrightnessValue[j];
        BrightnessPoints.push(x + "," + y);
        x += widthPerSample;
    }

    BrightnessElement = document.getElementById('brightnessGraph');
    updatedBrightnessPoints = BrightnessPoints.join(" ");

    BrightnessElement.setAttribute('points', updatedBrightnessPoints);
}

function displayValues() {
    document.getElementById('Brightness').innerHTML = "Current Brightness
Level:" + Light;
}
```

JavaScript Loop | Select Macro | select a device | Add Macro

```
// No need for a while loop, the JavaScript Loop section already loops!
await mSec(1000);
Light = analogIn(LightSensor);
BrightnessValue.push(Light);
updatedGraph();
displayValues();
```

Figure 9: Expect runlinc result screenshot

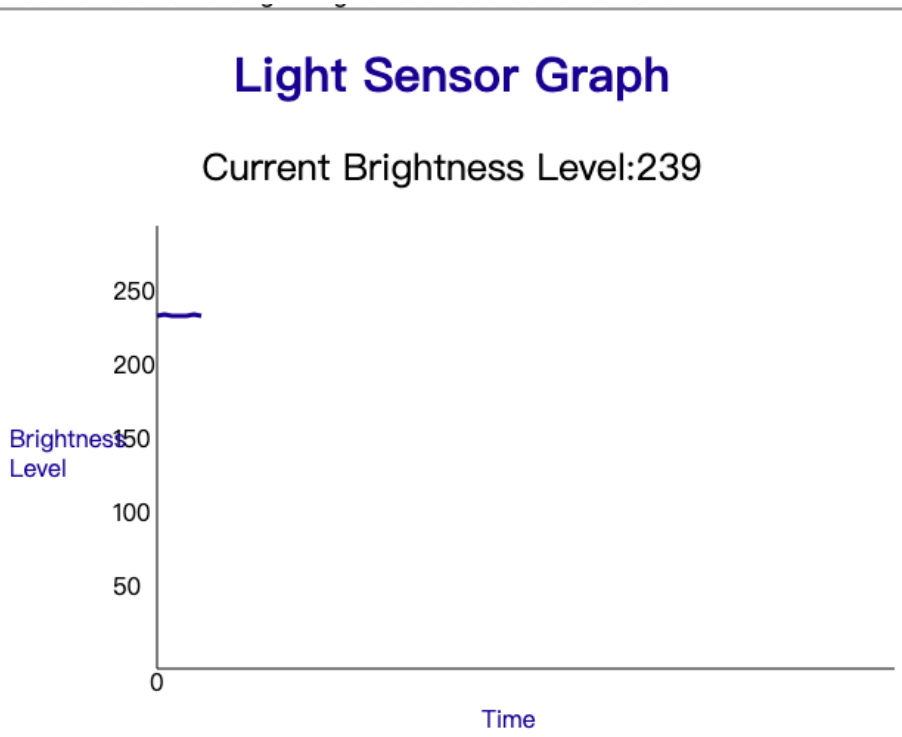

Figure 10: Expect webpage screenshot

Summary

Light is always playing a crucial role in the growth of green plants. Through monitoring the real-time brightness level of the growing environment of plants, farmers could control the light intensity to provide a more suitable growing environment for crops.