# Automatic Gate Lights using BOLT IoT

This tutorial can also be viewed at:-
https://projectsubmission.boltiot.com/?p=20478&preview=true

*When it is dark, the LED will glow automatically!*

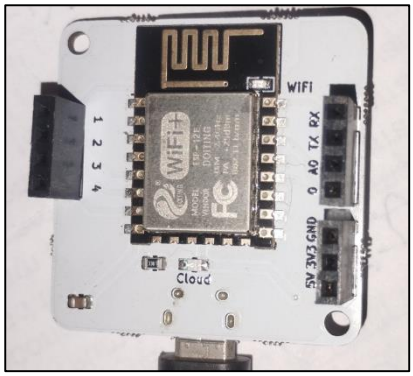
## Problem Instance

This project takes up a real world service and simplifies its implementation using BOLT IOT kit. We have seen how Automatic Gate Lights are installed in the exterior of house, to illuminate the surrounding after the sun sets. But what if you are not at home still want the exterior door lights to be lit at night? People do so in order to get proper illumination for their home CCTV camera, or for other reasons. The automatic Gate Lights can relieve them as it switches on automatically.

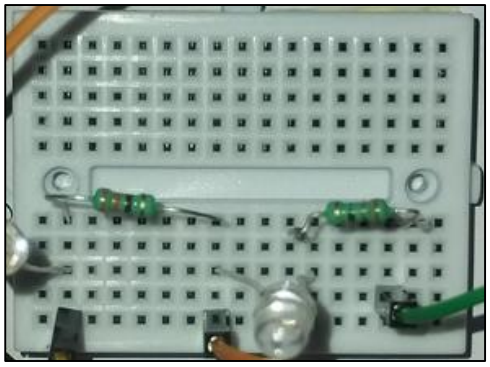
## Things used in this project

### Hardware Components

- Bolt IOT WiFi Module




- Mini Breadboard



- Jumper wires (at least three)
- LED (it is used to simulate the gate Light in this setup)

- LDR (for light sensor)

- Resistors (2 resistors)

All of the above hardware components are included in the BOLT IOT/ML training starter kit.

## Software, Apps and Online Services

- Bolt cloud (for API key and Device Id)

- Twilio (third party message delivery service)

- Python IDE and interpreter

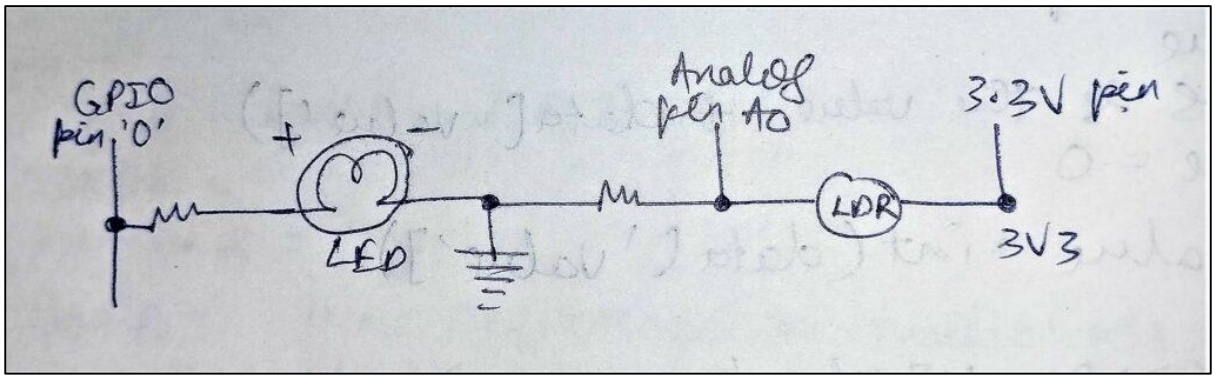

# Overview and objective

In this project, we shall create an automatic gate light system. For demonstration purposes, the gate light is being showcased by using an LED. The flow of the objective is as:-

1. The LDR senses the environment every minute and its reading is collected by software code.
2. If the intensity of light as collected by LDR is less than our minimum set threshold (that is defined as a criteria for darkness), then we switch on the gate Light as it denotes sun has set.
3. Thus our LED will be lit.
4. Why then not used a simple timer based approach to switch the LED on and off?
5. Because, in cloudy weather also, there may be need for light, and the timer based approach to switch the LED on or OFF simply does not work.
6. In every case, the LDR will sense the light and trigger the switching on of LED only when required.
7. When there is enough light, we shall automatically turn off the LED, if it is previously on.
8. If we have any system failure due to which LED can't be controlled automatically, we send SMS to inform the user.
9. An important catch here is to avoid a loop. Low intensity=>switch on LED=> Intensity increases=>Switch off LED=>Low intensity=>..... Read along to see how we take care of this.

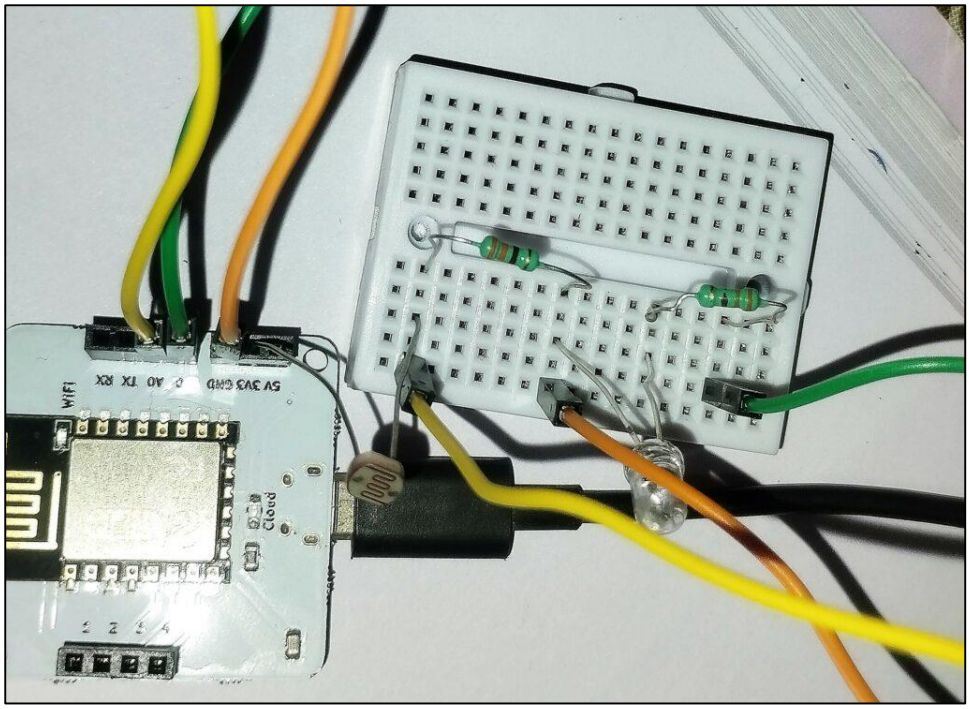
# Hardware Setup

Follow these steps to make the hardware circuit connection:-

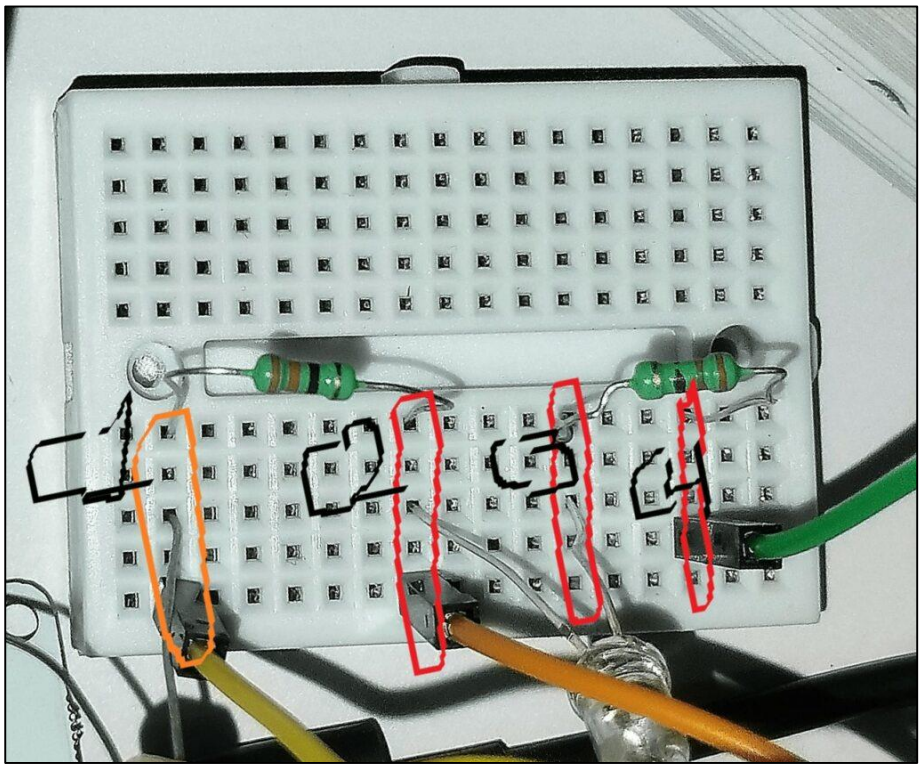



*Circuit connection for the project*

1. Make circuit connection as per the LDR circuit as taught in the course:-
    a. Connect one end of LDR to the “3v3” pin and other end in one of the column of the breadboard. Call this column c1.
    b. The columns on one side of mini breadboard are at equal potential. Connect a jumper wire from “A0” pin to the column c1. Thus other end of LDR is connected to “A0” pin.
    c. Then connect one leg of resistor to c1 column and other end to another column of breadboard, call it c2.
    d. Join a jumper wire from “GND” pin of BOLT and the column c2. This completes the LDR circuit.
2. Make the circuit connection for LED as taught in the course:-
    a. Connect the -ve terminal (shorter leg) of LED to the GND pin, i.e., the column c2 of breadboard. Connect longer (+ve) leg of LED to another column, say c3.
    b. Connect another resistor across column c3 and another column, say c4, of breadboard.
    c. Connect another jumper wire from column c4 to GPIO pin “0”. Thus the +ve pin of LED is connected to the GPIO pin 0. This completes the LED circuit.
3. The circuit connection made using hardware is shown.



*Note the color of the wire to understand the circuit. Black adapter is only to power up the bolt module, so ignore it.*




*Just for understanding c1,c2,c3,c4 as per the text*

# Software Programming

# Algorithm

1. Fetch the LDR sensor data. If the intensity is lower than the set threshold, and the LED is not ON, then
    a. Try to switch on the LED.
    b. If the LED can't be switched on due to network reasons, inform the user.
    c. Save the reading of low intensity at which LED was turned on.
2. If the reading of LDR is more than the set threshold, then
    a. If the LED is not ON then do nothing.
    b. But if the LED is ON, do not simply turn it off as it may trigger low intensity next time and cause looping of LED states to ON and OFF every minute.
    c. Check if current intensity fetched by sensor is a "limit" above the previous value when the LED was lit due to low intensity.
        i. This "limit" is defined by us, and is assumed the approximate increase in intensity due to turning on of LED.(This has to be tested several time to get it right)
    d. If it is, then probably the intensity will be higher than threshold even if LED is turned off. Hence turn the LED off, and if failed then message the user regardimg the issue.
    e. If the difference between current and previous sensor reading is less than the limit, then do not turn off the LED. This step avoids looping that we pointed out in objective.
3. Repeat steps 1 and 2 each minute (i.e. every 60s in an infinite while loop).

# Python code implementation

# conf.py

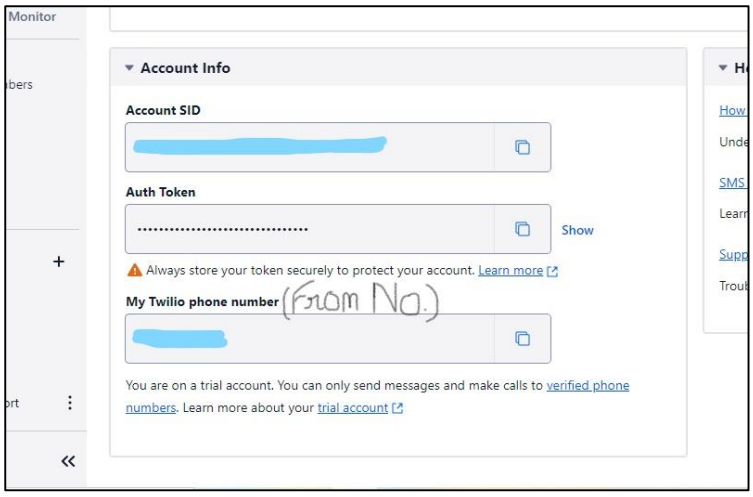
```
#Harsh Choudhary

bolt_api_key="xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
bolt_device_id="BOLTxxxxxx"

#twilio credentials
from_number="xxxxxxxxxxxx"
account_sid="xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
auth_token="xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
to_number="xxxxxxxxxxxxx"  #add your country code,ex:"+9158565"
```

# Twilio credentials

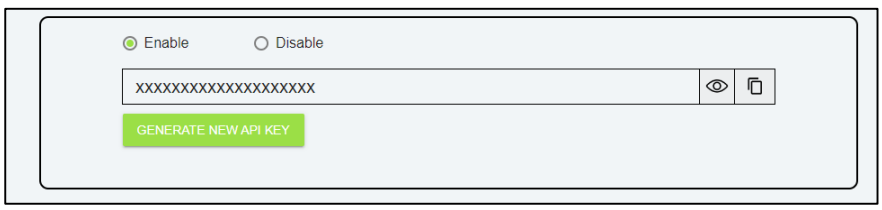
Log in to your Twilio account, then the credentials you need are available here:-




*The “to number” will be the one you used to create your Twilio account.*

# Bolt Credentials

Log in to the Bolt cloud and go to APIs tab.[https://cloud.boltiot.com/api]



## controller.py

```
#Harsh choudhary
import json,conf,time
from boltiot import Sms,Bolt

#set the thresholds for min_intensity below which the light must be turned on
min_intensity= 50

#set up the Bolt object and Sms instance
mybolt=Bolt(conf.bolt_api_key,conf.bolt_device_id)
mybolt_sms_test=Bolt("fhfjhjgkjhk",conf.bolt_device_id) #this is made to have a failing
api digital write so that sms service is demonstrated
sms=Sms(conf.account_sid,conf.auth_token,conf.to_number,conf.from_number)
led_on=False  #maintain the current state of LED
prev_intensity=-1 #only initialisation
limit=10    #this is the expected increase in intensity due to LED light
#we have to tune limit properly

while True:
   #get the sensor value
   #the pin is A0
   reading=mybolt.analogRead('A0')
   data=json.loads(reading)
   print(data)
   try:
      sensor_value=int(data['value'])
      #test to see if the light intensity is less than threshold
      if(sensor_value<min_intensity):
         #thus switch on the LED
         print("Must switch LED on")
         if(not led_on):
            #if the led is already on then we can not help the situation
            #trying to switch the LED on
            print("Led is previously off")
            #command=mybolt_sms_test.digitalWrite('0','HIGH')
             #The above commented command is left as is to uncomment and test the
sms delivery, as it leads to exception
            command=mybolt.digitalWrite('0','HIGH')
            print(command)
            command_data=json.loads(command)
```

```
            if(command_data['success']==0):
                print("Could not switch the LED ON")
                #send sms to the owner
                message_response=sms.send_sms("There is some defect in LED or system.
It is dark. Replace the LED")
                print(message_response)
            else:
                #led has been switched on
                prev_intensity=sensor_value
                led_on=True
        else:
            #the environment is not dark, so if the LED is ON, switch it off
            #it does no harm to switch off the LED even if it is off
            #but we wish to inform the house owner if despite our attempt LED could not
be turned off
            #the owner may then manually turn off the LED
            if(led_on):
                #if prev_intensity is -1 then led_on must be false itself
                #when led_on is set True, prev_intensity can not be -1 as it is also updated
together
                if(sensor_value-prev_intensity>limit):
                    #only in this case switch off the LED
                    #because otherwise the increase in intensity due to the LED causes the
sensor to detect more light and turn LED off
                    #which would then make LDR detect low light
                    #this leads to looping
                    command=mybolt.digitalWrite('0','LOW')
                    print(command)
                    command_data=json.loads(command)
                    if(command_data["success"]==0):
                        print("Will try to switch LED off a minute later")
                        #inform the owner
                        message_response=sms.send_sms("There is some defect in system. The
LED needs to be turned off.")
                        print(message_response)
                    else:
                        led_on=False

    except:
        print("Can't read sensor data")

    time.sleep(60)
```

# The output we get

When we run the controller.py file, we see that the LED responds appropriately with respect to the environment lighting as captured by the LDR. Here are the photos for a quick glance at the results:-

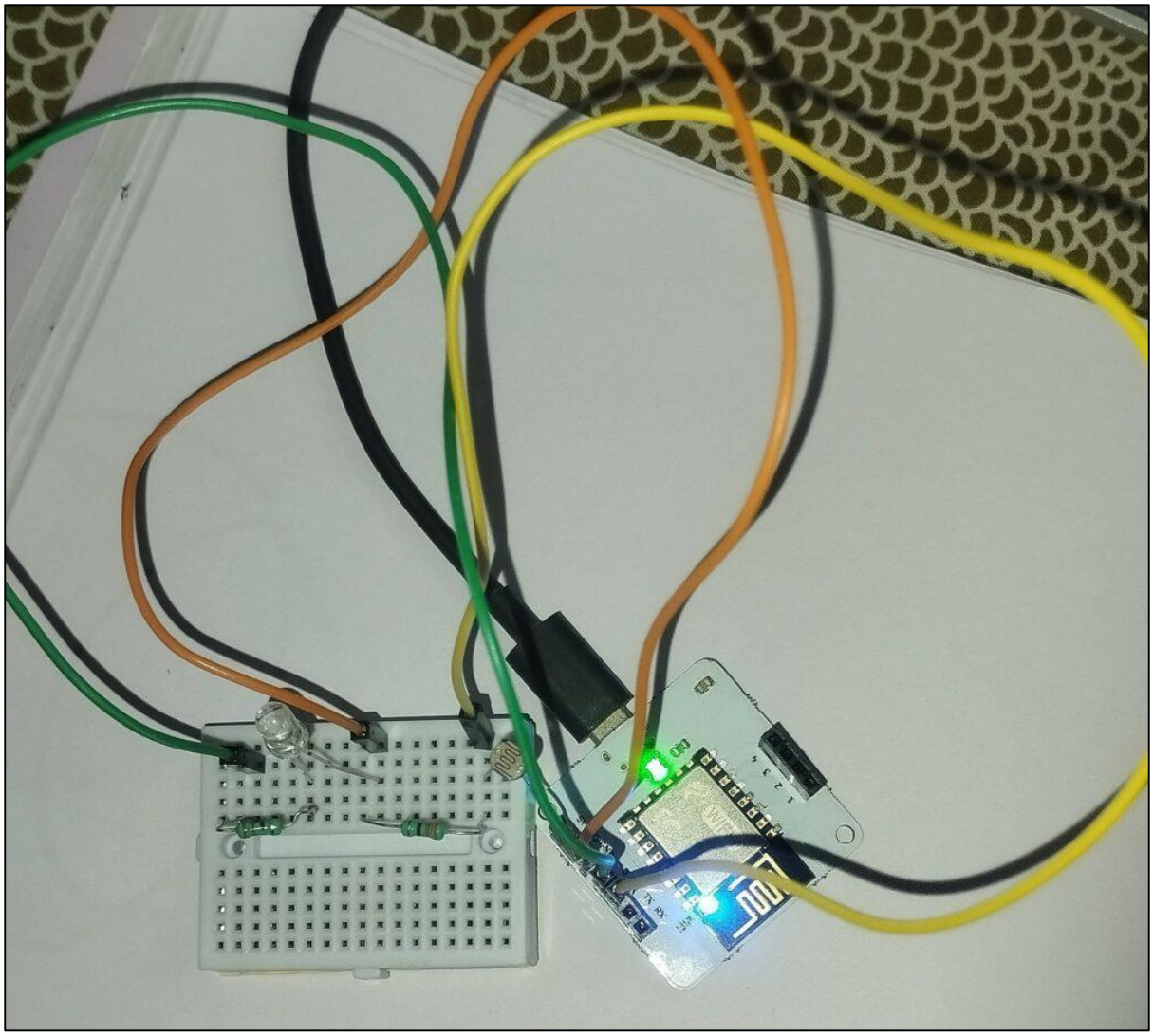
```
{'value': 'Device is offline', 'success': 0}
Can't read sensor data
{'value': 'Device is offline', 'success': 0}
Can't read sensor data
```

*When device is offline the code handles this exception and does not crash*

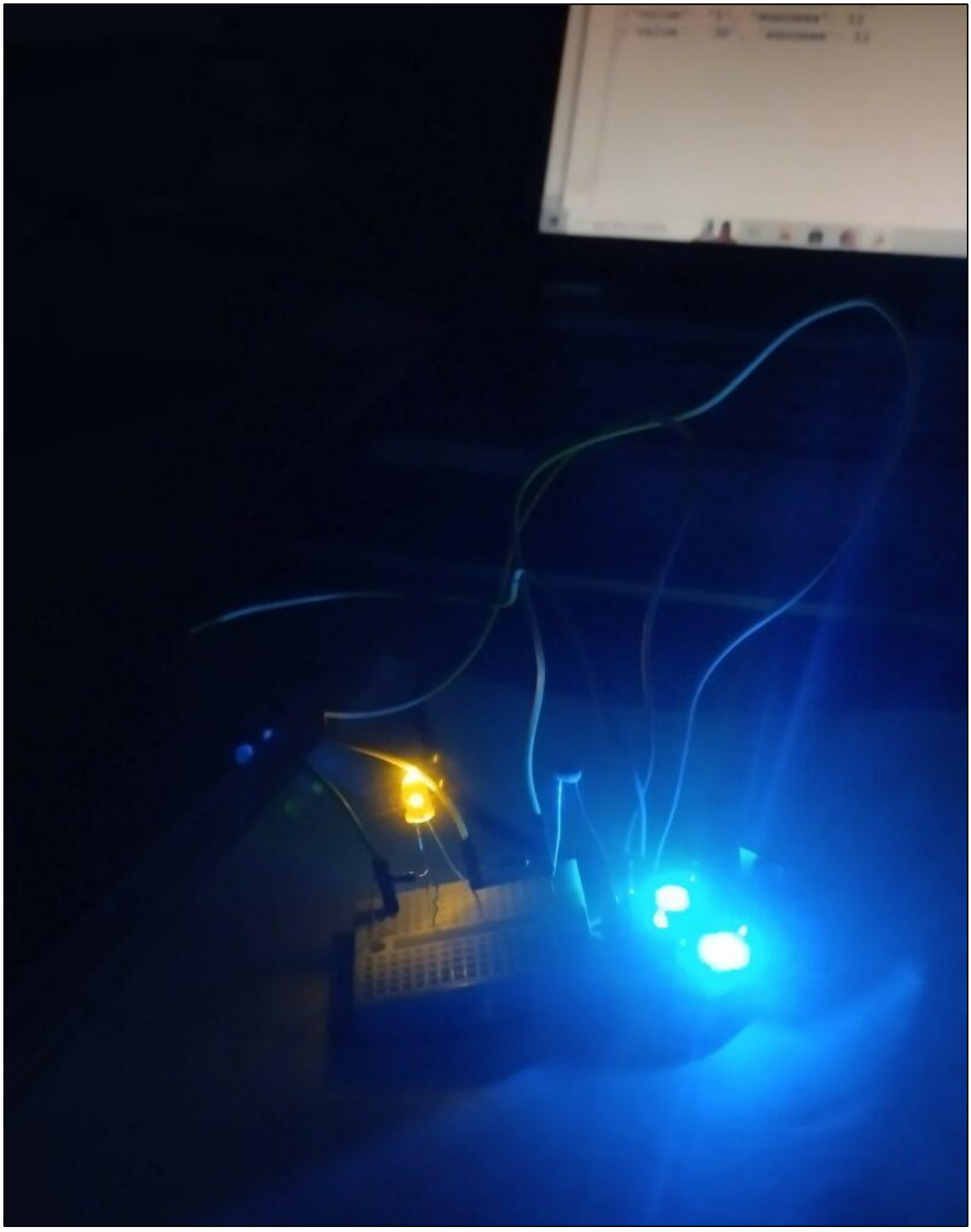
```
{"value": "1", "success": 1}
{'value': '67', 'success': 1}
{'value': '68', 'success': 1}
{'value': '15', 'success': 1}
{"value": "1", "success": 1}
{'value': '30', 'success': 1}
{'value': '86', 'success': 1}
{"value": "1", "success": 1}
{'value': '72', 'success': 1}
{'value': '72', 'success': 1}
{'value': '15', 'success': 1}
{"value": "1", "success": 1}
```

*This is the format of the response we receive from the BOLT device*

We kept the min intensity value to 50, as our room had appropriate lighting upto intensity 70. So we shall switch the lights off to demonstrate.



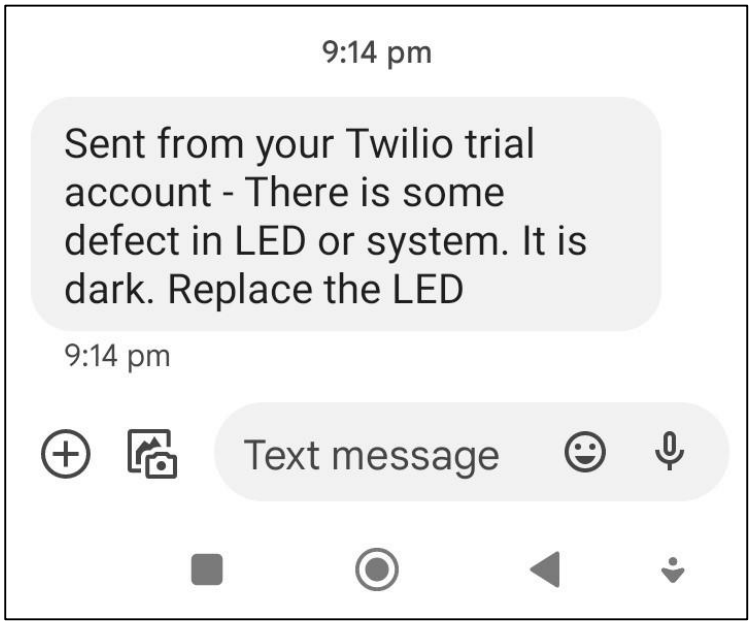
*In proper lightning conditions the LED is off.*



*When we make the room dark by switching off every light, the LED is switched on within 1 minute as the sensor collects the current intensity every minute.*

```
{'value': '74', 'success': 1}
Must switch LED on
Led is previously off
{"value": "Invalid API key", "success": 0}
Could not switch the LED ON
<Twilio.Api.V2010.MessageInstance account_sid=ACc97908c6b46784cf8da730ab28d759d7
 sid=SM3ebed798b5aaec6ea15776313bca23f9>
```

*The task of switching LED on failed when we use the wrong API key to do digital write.*




*This is the message received when the LED could not be switched on due to network issue.*

The video demo is available on You Tube [https://youtube.com/shorts/GKN4U7qEOkI?feature=share]

and also provided in the drive folder.

# Conclusion

Thus, we are able to build up a system where the LED switches on automatically when the surrounding is dark enough. In real world we can replace LED with a proper gate light that would form this “automatic gate light”- whenever the surrounding gets dark the light switches on.

Nowadays this is implemented by installing scheduled automatic gate lights but this integration with LDR to capture intensity suits other needs such as turning the light ON dynamically whenever it is dark either due to clouds or due to night.