PR1101: Automation Project

SMART UNIVERSITY

PREPARED BY

Garv Baheti (2020BTechCSE031)

Shubham Sharma (2020BTechCSE072)

Siddharth Jangid (2020BTechCSE074)

FACULTY GUIDE

Mr. Divanshu Jain

Dr. Hanuman Prasad Agrawal

Department of Computer Science and Engineering

Institute of Engineering and Technology

JK Lakshmipat University Jaipur

November 2022

CERTIFICATE

This is to certify that the project work entitled "**Smart University**" submitted by **Garv Baheti, Shubham Sharma, Siddharth Jangid** towards the partial fulfilment of the requirements for the degree of **Bachelor of Technology in Computer Science Engineering** of JK Lakshmipat University, Jaipur is the record of work carried out by them under our supervision and guidance. In our opinion, the submitted work hasreached a level required for being accepted for the final submission of project.

----------	----------
Dr. Hanuman Prasad Agrawal	Mr. Divanshu Jain
Assistant Professor	Assistant Professor
Department of Computer Science Engineering	Department of Computer Science Engineering
Institute of Engineering & Technology	Institute of Engineering & Technology
JK Lakshmipat University, Jaipur	JK Lakshmipat University, Jaipur

Date of Submission:

ACKNOWLEDGEMENTS

We would like to express our appreciation to our faculty and advisors, **Dr. Hanuman Prasad Agrawal**, Assistant Professor, IET, JK Lakshmipat University, **Mr. Divanshu Jain**, Assistant Professor, IET, JK Lakshmipat University, and **Dr. Devika Kataria**, Associate Professor, IET JK Lakshmipat University for their invaluable guidance, support and expertise throughout this project.

We would also like to extend our appreciation to **Dr Dheeraj Sanghi**, Vice Chancellor, JK Lakshmipat University, and **Dr Sanjay Goel**, Director, Institute of Engineering and Technology, JK Lakshmipat University for their ongoing encouragement and support during this project.

We would also like to thank our parents for their encouragement and support throughout the project.

Last but not least, we would like to thank our family and friends for their moral and emotional support throughout this project.

Sincerely yours,
Garv Baheti
Shubham Sharma
Siddharth Jangid

ABSTRACT

JKLU has a lovely campus that draws many visitors. Campus management teams have made numerous measures to maintain the campus, however everything is currently handled manually, including manually turning on/off pole and fog lights on/off, irrigating the central lawn, and monitoring corridor lights. Our goal is to make the campus smart, with everything managed by IOT devices that may be operated manually or automatically. As soon as the sunlight goes under the visible region of our eyes this system will turn on pole and fog can be controlled via phone also, corridor lights will turn on for a few minutes when they detect motion, water pump will turn on automatically when soil moisture level is low while sending information on server, and fountains will be installed around the central lawn that will turn on when people walk around it.

CONTENTS

DOOR ACCESS CONTROL USING FACE RECOGNITION

CHAPTER 1: PROBLEM STATEMENT

The goal of this project is to design and implement an IOT system to make JKLU's campus more efficient and sustainable by automating manual processes, such as turning on and off pole and fog lights, irrigating the central lawn, and monitoring corridor lights.

CHAPTER 2: LITERATURE SURVEY

The implementation of Internet of Things (IoT) technology on university campuses is a growing trend in the field of smart campus management. Researchers have explored how IoT devices can be used to automate a variety of tasks, such as automating the turning on and off of lights, monitoring corridor lights, controlling irrigation systems, and more. In particular, studies have explored how IoT technology can be used to create a "smart campus" that is capable of providing a safe and secure environment for students and faculty. [1]

One study investigated the potential of using smart sensors to automate tasks such as turning on and off pole and fog lights, irrigation of the central lawn, and monitoring corridor lights. The study found that such automation can improve energy efficiency, reduce operational costs, and increase safety on campus [2]. Additionally, the study found that by using IoT technology, university staff can remotely control and monitor the system, as well as receive notifications about any potential problems.

Automatic irrigation systems are an important component of efficient and sustainable agricultural production, as they help reduce waste while also providing a reliable source of water for crops. Automatic irrigation systems employ technologies such as sensors, controllers, and pumps to monitor and regulate water supplies. These systems can be self-learning and adaptive, capable of adjusting water supply based on actual need, weather conditions, and other factors. Eventually such systems would be integrated into a university's overall smart campus infrastructure. The automatic irrigation systems are an important technology for smart university applications, as they enable optimal use of resources and improved food production. Research in the field of automatic irrigation systems has become increasingly important and numerous studies have focused on designing and optimizing these systems. New advancements in IoT technologies, machine learning, and energy saving methods have served to increase the potential of automated irrigation systems in this setting and are likely to become increasingly important in the future.

Nowadays, with the growth of technological progresses, many research studies and applications have been conducted in order to develop a smart university system. Recently, the concept of Smart University has been widely studied and numerous approaches have been

proposed to create a collection of intelligent tools. Among them, motion controlled lighting is one of the promising topics for smart university implementation. In this literature survey, the state-of-the-art research studies related to motion controlled lights for their implementation in Smart University will be discussed.

An interesting approach by Kaleni-Ahmad, H., et al. (2019) involves the usage of motion sensors for developing a smart energy saving solution in college dorms. The study utilizes occupancy sensor-equipped lighting control systems (OLC) to control the indoor light regulators and reduce energy waste. The proposed system also enables the students to control the temperature, humidity and illumination of the room. The authors proposed an energy model to show the efficiency of the proposed system and the results demonstrated a reduction of 23.15% energy consumption when the proposed system is enabled.

Sun, X., et al. (2018) proposed an indoor lighting control system based on an extended hierarchical fuzzy controller combined with a three-level feedback. The study used three types of sensor to control the lighting status - a motion sensor to detect movement, a light sensor to measure the intensity of light, and a temperature sensor to sense the temperature. The experimental results showed that the proposed system had a good performance for controlling indoor lightings.

Li, F., et al. (2018) also proposed an occupancy-Based smart lighting control system, consisting of Raspberry Pi, a PIR motion sensor, an LED light and a Bluetooth module. The system is able to detect and adjust the lighting status according to the occupancy of the room, and turn off the lights after an adjustable environment-configured time period. The proposed system also achieves energy savings.

Subramaniam, N. & Dubey, V. N. (2017) presented an effective indoor lighting control system that is used to maintain the energy consumption in university buildings. The system is based on a light sensor, a humidity sensor, and a temperature sensor that are connected to a Raspberry Pi computer. The proposed system utilizes a PIR sensor to detect the presence of human beings in the premises and the system adjusts the light intensity depending upon the occupancy of the room.

Therefore, the use of motion controlled lighting in Smart University is an efficient and

practical approach to save energy. Motion sensors can be used to detect the presence of people and adjust the light intensity depending upon the occupancy of the room. This reduces the energy consumption as well as enhances the comfort and security level of the students in the university. Additionally, the integration of accessories into the control system like light sensors, humidity sensors, and temperature sensors make the system more efficient and user-friendly.

Other studies have focused on the potential of using IoT technology to create a "smart campus" that can be used to provide a more interactive and engaging campus experience. For example, one study explored the potential of using IoT devices such as sensors and cameras to create interactive fountains [3] that turn on when people walk around them. Additionally, the study explored how IoT technology can be used to create a "smart campus" that can be used to provide a more interactive and engaging environment for students and faculty.

In conclusion, research has shown that the implementation of IoT technology can be a useful tool for creating a "smart campus" that can provide a safe and secure environment for students and faculty, while also providing a more interactive and engaging experience. Further research is needed to explore the potential of using IoT technology to create a "smart campus" that can provide a safe and secure environment for students and faculty [4].

Blynk is an open-source IoT platform of Mobile Applications and Server that enable users to control home appliances and devices with their smartphone. It supports a wide range of devices, ranging from embedded micro-controllers to mobile phones. The platform provides a user-friendly interface which makes it easy to connect and control devices.

An extensive study of the existing research papers on Blynk app for IOT has been done by the authors in this article. First, the paper analyzed various technologies used in the development of Blynk. These include communication protocols, security measures, and data management. The paper then reviewed some of the existing IOT applications that have been developed using Blynk, including controlling drones and robots, home energy management, monitoring indoor air quality, and smart home automation. It further discussed the performance evaluation of the system, including latency and throughput measurements.

The paper further discussed various challenges faced by the developers of the Blynk app for

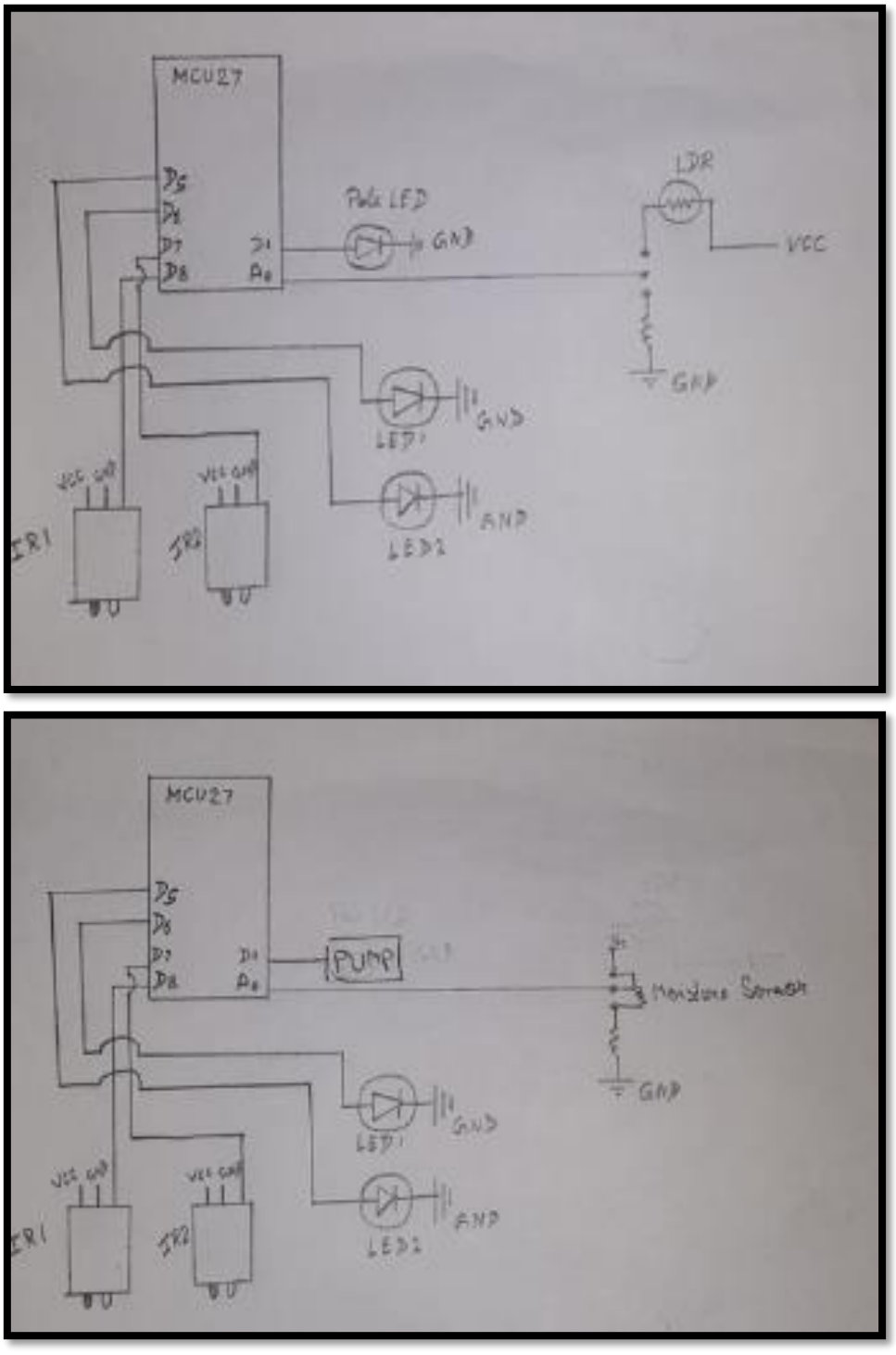
IOT, such as data privacy, scalability, and resource utilization. Additionally, various security measures such as authorization, authentication, and encryption have been identified as a potential solution for safeguarding the system against potential cyberattacks. Finally, the paper discussed some open research topics that need to be addressed in order to improve the implementation of the Blynk app for IOT such as optimization of the application server, implementation of standard APIs, and integration of external libraries.

CHAPTER 3: PROJECT OBJECTIVES

We would use this project to make our daily lives easier, this project is to create an IOT system for JKLU's campus that automates the management of the campus,

- Automatic Light System
- Motion Controlled Lights in corridor
- Auto irrigation system for central lawn
- Automatic fountain system
- Complete control on the system through Blynk App

CHAPTER 4: SYSTEMATIC DIAGRAM



CHAPTER 5: COMPONENTS USED

The materials required for this project are:

- **NodeMCU**

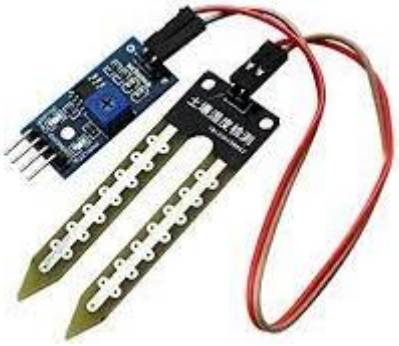
NodeMCU is an open-source IoT platform that uses an ESP8266 Wi-Fi module to connect to a variety of sensors and electronic components. It can be programmed with the Arduino IDE, allowing users to create IoT projects with ease. NodeMCU is used for creating applications ranging from home automation to wearables and beyond.

- **IR Sensor**

An infrared (IR) sensor is a device that detects or measures infrared radiation. It is typically used to detect the presence of people or objects in a particular area, measure their temperature, or detect motion. Infrared sensors are also used in some types of burglar alarms, and in many security systems.

- **Moisture Sensor**

A moisture sensor is a device used to measure the amount of water present in a given substance, such as soil, plants, food, or other materials. Moisture sensors can be used to detect the presence of water, monitor changes in moisture levels, and measure water flow. The most common type of moisture sensor uses electrical resistance to measure the amount of moisture in the material. Other types of sensors use capacitance, ultrasonic waves, or microwave radiation to measure the moisture content.



- **Water Pump**

A water pump is a device that moves water from one place to another using mechanical or thermal energy. In most cases, water pumps are powered by electricity, but they can also be powered by gasoline, diesel, or other forms of energy. Water pumps are used in a variety of applications, including transferring water from a well or reservoir to a home, spraying water in agriculture, irrigating fields, and supplying water to firefighting systems.

- **LDR Sensor**

LDR (light dependent resistor) sensors are sensors that measure the intensity of light. They work by using a resistor that changes its resistance depending on the amount of light that it is

exposed to. When the light is low, the resistance of the sensor is high, and when the light is high, the resistance of the sensor is low. LDR sensors are commonly used in applications such as light dimmers, light switches, and automatic night lights.

- **Fountain**

A fountain is a structure or device which propels a jet of water into the air, typically for decoration or for providing drinking water. Fountains can be found in public parks and public squares, in front of buildings, or in private gardens. Fountains come in a wide variety of designs and styles, ranging from simple gravity-powered structures to complex, modern-day designs that use pumps and motors to create spectacular water displays.

- **Relay**

A relay is an electrical switch that opens and closes circuits electromechanically or electronically. It consists of an electromagnet that is actuated by an electric current. When the current flows through the coil, the resulting magnetic field attracts a contact that completes the circuit. Relays are used to control a high-power circuit from a low-power signal, as in a car's starter solenoid.

CHAPTER 6: SOFTWARE LIBRARIES USED

Include ESP8266 Core to Arduino IDE

1) Goto 'Preferences' and enter the following URL to Additional Board Manager URLs

```
http://arduino.esp8266.com/stable/package_esp8266com_index.json
```

2) Open the Boards Manager (*Tools > Board Menu*)

3) Search for "esp8266" and install the latest version

4) Select your board under *Tools > Board* and define Baud Rate etc.

Install Blynk libraries

1) Install the latest release of the Blynk libraries on GitHub

2) Unpack it

3) Move the libraries to *C:/User//Documents/Arduino/libraries*

Install Blynk App

1) Download the App for iOS or Android

CHAPTER 7: METHODOLOGY

Connecting Blynk with NodeMCU:

1. Connect an ESP8266 board (NodeMCU) to your computer using a USB cable.
2. Open the Arduino IDE, select Tools->Board->NodeMCU and then select the COM port.
3. Install the Blynk library from the Arduino IDE library manager.
4. Select File -> Examples -> Blynk -> Boards_WiFi -> NodeMCU.
5. Download the Blynk app and create an account.
6. Create a new project in the app and select NodeMCU as the device.
7. Get the Auth Token associated with the project and copy it.
8. Paste the Auth Token into the example code in the Arduino IDE.
9. Connect the NodeMCU to the WiFi network.
10. Verify and upload the sketch to the NodeMCU.
11. The NodeMCU will now be connected to the Blynk server.
12. Verify that the NodeMCU has successfully connected to the Blynk server by checking the serial terminal.

Automatic Light System:

- Set Up the Blynk Application:
    a. Download and install the Blynk app from the App Store or Google Play Store.
    b. Create an account and add a new project.
    c. Choose your hardware, and add a suitable widget.

- Connect NodeMCU with Blynk:
    a. Download and install the Blynk Library.
    b. Connect the NodeMCU with the LDR.
    c. Create a Blynk Auth Token, and add it to your Sketch.

- Program NodeMCU with LDR Sensor:
    a. Create an NodeMCU Sketch for your project.
    b. Set up the LDR Sensor to detect light levels and send this data to the NodeMCU.
    c. Program the NodeMCU to read the LDR data and to turn on or off the relay.

- Test Your System:

a. Connect the relay to the light bulb.
b. Test the system to ensure that the light responds to the LDR data.
c. Test the system with the Blynk app.
d. Update and enhance your system, as needed.

Motion Controlled Lights in Corridor:

1.Set up your NodeMCU: Install the Arduino IDE software on your computer and then install the NodeMCU Board Library. If your NodeMCU Board is configured successfully, connect it to your computer and upload a sketch to ensure that everything is set up correctly.

2.Connect your IR Sensor and NodeMCU: Once you have the IR Sensor and the NodeMCU board, connect the one of the NodeMCU digital pins to the output pin of the IR Sensor. Make sure that the power pins and ground pins are connected correctly.

3.Create a Blynk Project: Create a new Blynk project in the Blynk mobile app. Connect it to your NodeMCU board over Wi-Fi. Create a virtual switch and couple it with one of the digital pins of the NodeMCU board.

4.Program your NodeMCU: Program your NodeMCU board with the necessary sketch. The sketch should contain the logic to turn on an LED when the same digital pin of the NodeMCU board is activated. To activate the digital pin, use the output from the IR Sensor.

5.Control Motion Controlled Lights: Once everything is set up, you can control your Motion Controlled Lights with the virtual switch in the Blynk app. Whenever the IR Sensor detects motion, the LED will be activated. You can adjust the settings in the app to control the amount of motion that is detected by the IR Sensor.

Fountain System:

1. Set up the Blynk app:

Begin by downloading and installing the Blynk app for your mobile device. Create a new project and select NodeMCU as your hardware. After that add a button widget to the project.

2. Connect the NodeMCU to the Blynk app:

Set the WiFi credentials in the NodeMCU in order to connect it to the Blynk app. In the Blynk app, click on the Settings tab and click on “Connect NodeMCU”. Enter the same WiFi credentials and your NodeMCU should now be able to connect to the Blynk app.

3. Set up the motion sensor:

Connect the motion sensor to the NodeMCU. Then add the following code to the NodeMCU to enable the motion sensor to detect motion or movement.

4. Setup the fountain system:

The next step is to build the fountain system which consists of a fountain pump, water reservoir, and hose. The pump is connected to the NodeMCU with a relay and when the motion is detected, the relay triggers the pump and water will be pumped through the hose.

5. Design and assembly:

Once all the components of the fountain system are established, you will then need to design and assembly the entire system. You will need to design a base for the pump, water reservoir, and the hose. Make sure the pump is securely attached to the base and the hose is connected properly.

6. Connect the system to the Blynk app:

Once the system is setup, you will then need to add the code to the NodeMCU that enables the system to be connected to the Blynk app. When the button is pressed on the Blynk app, it will trigger the circuit and the fountain system will activate.

7. Test and refine:

Run a series of tests to make sure that the system works as intended. If you encounter any issues, make necessary adjustments to the code or the hardware until the system is working properly.

Automatic Irrigation System:

Step 1: Set up and Configure Node MCU:

Begin by setting up your NodeMCU. Connect it to your WiFi router and give it the

necessary permissions.

Step 2: Install the Blynk and Moisture Sensor Libraries:
You will need to install the Blynk and Moisture Sensor libraries to your NodeMCU so it can communicate with the cloud and the sensor.

Step 3: Set up Virtual Pin for Blynk:
Create a virtual pin in Blynk and assign it to your NodeMCU. This will enable them to communicate with each other.

Step 4: Connect your Moisture Sensor to the NodeMCU:
Connect the moisture sensor to the NodeMCU according to the instructions provided.

Step 5:Create the Program for the NodeMCU:
Now it's time to create a program for the NodeMCU. This program should read the moisture level from the sensor and compare it to a predetermined value. If the value is below the desired value, it should turn on the pump and send a live moisture level status to Blynk.

Step 6: Test and Debug the Program:
Before deploying your program, it's important to test and debug your program to make sure everything works as expected.

Step 7: Deploy the Program:
Once you've tested and debugged your program, you can deploy it to the NodeMCU. In order to make sure it's working properly, monitor the live moisture levels on Blynk to make sure the readings are accurate.

Step 8: Operate your Automatic Irrigation System:
With the program deployed, your automatic irrigation system should be ready for operation. Monitor the moisture levels and make sure the system is running properly.

CHAPTER 8: RESULTS

Model Photo

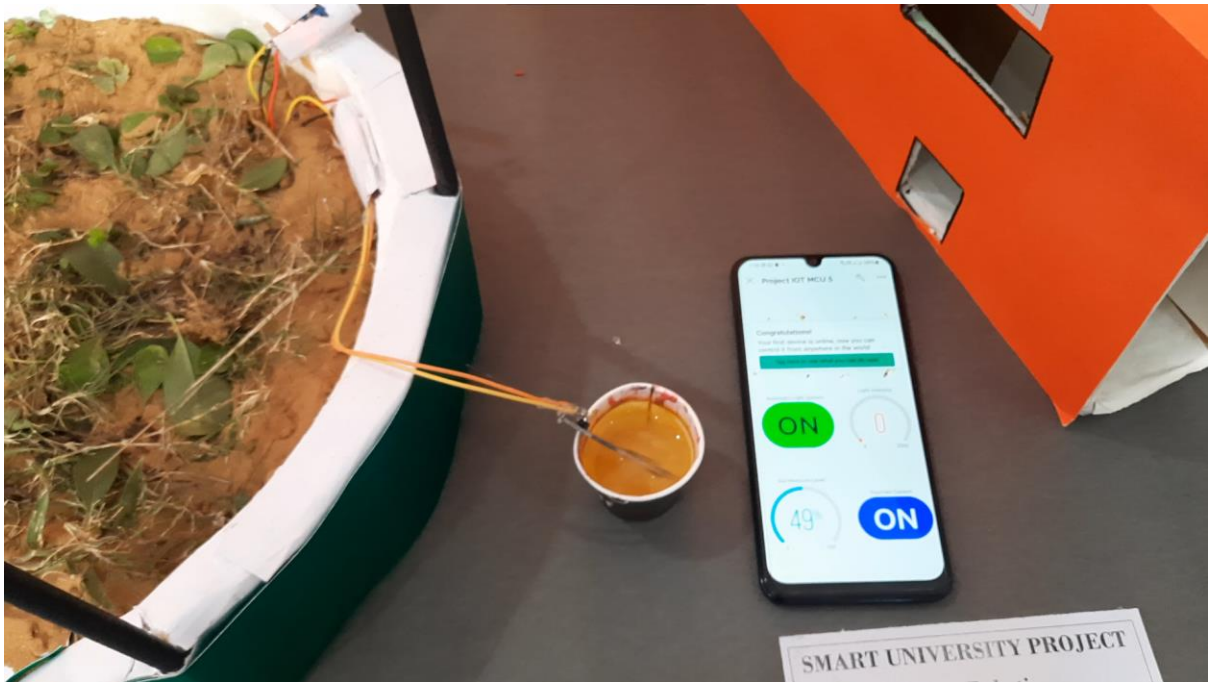
AUTOMATIC LIGHT SYSTEM

LDR Sensor on top of LRC to check Light Intensity

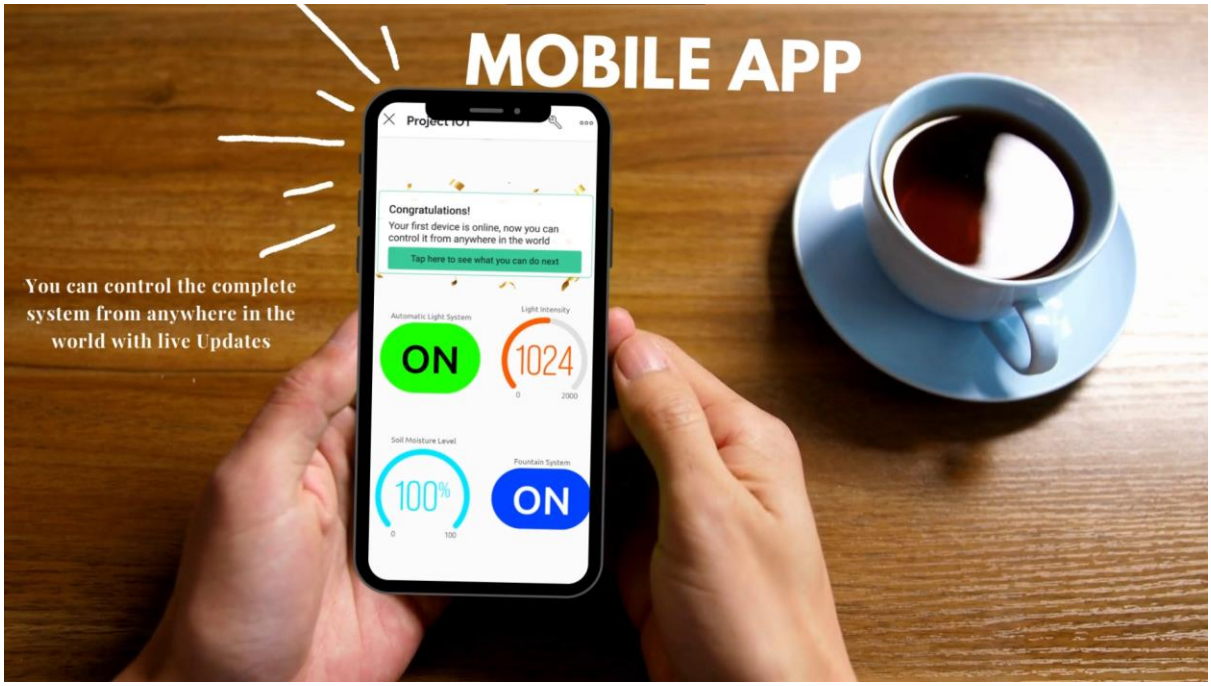
Pole LED

Motion controlled LEDs in Corridor

FOUNTAIN SYSTEM & AUTOMATIC IRRIGATION SYSTEM


ON
49
ON
SMART UNIVERSITY PROJECT

MOBILE APP


MOBILE APP
You can control the complete system from anywhere in the world with live Updates
Congratulations!
Your first device is online, now you can control it from anywhere in the world
Tap here to see what you can do next
Automatic Light System
ON
Light Intensity
1024
Soil Moisture Level
100%
Fountain System
ON

CHAPTER 9: LEARNING OUTCOMES

Through this project, we have learnt about the following:

- Sending and receiving the data of sensors to a cloud service/database for easy accessibility.
- Gain an understanding of the fundamentals of Internet of Things (IoT) and its technologies.
- Learn to design, program, and build computerized systems using IoT boards and sensors.
- Develop the ability to implement safe and secure designs for automatic lighting systems, motion-controlled lights and fountains, and automatic irrigation systems.
- Enhance skills in configuring and controlling IoT devices with the Blynk app.
- Learn to build a smart university project with artificial intelligence and big data analytics.
- Acquire practical experience troubleshooting, maintaining, and upgrading IoT-enabled systems.
- Demonstrate knowledge of performance optimization strategies for IoT systems.
- Appreciate how a project like the “Smart University” can improve efficiency and convenience.

REFERENCES

1) Haneef, M., Alvi, A., & Ahmed, I. (2019). Automated control of fountains using sensors and actuators. International Journal of Innovative Technology and Exploring Engineering, 8(5), 2296-2303.
2) Kaur, S., Verma, M., & Singh, R. (2020). Automated lighting control system for corridors using IoT. International Journal of Innovative Technology and Exploring Engineering, 9(4), 637-642.
3) Kaur, S., Verma, M., & Singh, R. (2019). Automated control system for dimming streetlights using internet of things. International Journal of Innovative Technology and Exploring Engineering, 8(6), 2456-2460.
4) Patel, A., Chaudhari, A., & Pandya, A. (2020). Automatic irrigation system using IoT. International Journal of Innovative Technology and Exploring Engineering, 9(1), 812-816.
5) V. Abdelaziz, A. McIntosh, A. Alshare, and S. Vyosky, “Blynk: An IOT application platform for users, developers and makers,” 2016 International Conference on Information and Automation for Sustainability (ICIAfS), 2016.
6) F. Rossi, G.-A. Mihailescu, and T.-A. Dinu, “IOT applications development with Blynk: Controlling robots and drones,” 2016 International Conference on Automation, Robotics and Applications (ICARA), 2016.
7) A. P. Raymond, G. Wilson, and F. Kojus, "Applying Blynk to indoor air quality monitoring: an IOT approach," in Proceedings of the 4th International Conference on Computers, Communications, and Control Technologies (IC4T 2017), 2017, pp. 123-128.
8) B. Horton, C. Thompson, M. Quinn, and J. Ramirez, "Using Blynk for energy resource management: An IOT approach," in Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI), 2017.
9) D. Bindal and D. Singh, "Secure and Robust IOT Solution with Blynk," in International Conference on Innovative Research in Computer Science & Engineering, 2018.
10) Ida, C. (2020). Automatic Irrigation System Using Internet of Things. International Journal of Engineering and Advanced Technology (IJEAT), 9(6), 1164–1168.

11) Klemens, R. (2005). Energy-saving automatic irrigation systems. CIE Newsletter, 19(3), 42-57.

12) Ismail, K. M., & Habib, A. M. (2011). Automatic irrigation system intelligent control application. International Journal of Computer Applications, 24(11).

13) Shoaib, M., Mohammad, A., & Irfan, M. (2019). Design and Development of Intelligent Automatic Irrigation System Using Arduino Uno and TGS-8002. International Journal of Advanced Engineering Research and Science (IJAERS), 6(5), 164-166.

14) Barreto, B. M., & Lee, S. S. (2011). Automated control of irrigation systems using fuzzy logic. Computers and Electronics in Agriculture, 76(2), 315-320.

15) Wang, C. H., Peng, C. M., & Liu, Y. T. (2011). Fuzzy model for automatic irrigation system. Expert Systems with Applications, 38(5), 5272–5278.

APPENDIX

Code:

```
/*************************************************************

  RUN ON NODE MCU NUMBER 27
 *************************************************************/

// Template ID, Device Name and Auth Token are provided by the Blynk.Cloud
// See the Device Info tab, or Template settings
#define BLYNK_TEMPLATE_ID           "TMPLEzG0Eco2"
#define BLYNK_DEVICE_NAME           "Quickstart Device"
#define BLYNK_AUTH_TOKEN            "kyYB7YW39DGv50pQVfWJioWz8hDEgWbU"


// Comment this out to disable prints and save space
#define BLYNK_PRINT Serial


#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

char auth[] = BLYNK_AUTH_TOKEN;

// Your WiFi credentials.
// Set password to "" for open networks.
//char ssid[] = "Baheti";
//char pass[] = "P@SSW0RD123";
char ssid[] = "me.baheti.07";
char pass[] = "123456789";
WidgetBridge bridge1(V8);

const int ledPin = 5;
const int ldrPin = A0;
const int IR1ledPin = 12; // choose pin for the LED
const int IR1inputPin = 13; // choose input pin (for Infrared sensor)
const int IR2ledPin = 14;
const int IR2inputPin = 15;
const int lrc_led = 4;

int ldrStatus = 0;
int IR1val = 0; // variable for reading the pin status
int IR2val = 0;
int value = 0;
// Timer: Auxiliary variables
unsigned long now = millis();
unsigned long lastTrigger1 = 0;
boolean startTimer1 = false;

unsigned long lastTrigger2 = 0;
boolean startTimer2 = false;

BlynkTimer timer;
```

```
// This function is called every time the Virtual Pin 0 state changes
BLYNK_WRITE(V0)
{
  // Set incoming value from pin V0 to a variable
  value = param.asInt();

  // Update state
//  Blynk.virtualWrite(V1, value);
}
// This function is called every time the device is connected to the
Blynk.Cloud
BLYNK_CONNECTED()
{
  bridge1.setAuthToken("xjAqVA6phIPFja-f8bjY5XzKwTqBl6z0");
  // Change Web Link Button message to "Congratulations!"
  Blynk.setProperty(V3, "offImageUrl", "https://static-
image.nyc3.cdn.digitaloceanspaces.com/general/fte/congratulations.png");
  Blynk.setProperty(V3, "onImageUrl",  "https://static-
image.nyc3.cdn.digitaloceanspaces.com/general/fte/congratulations_pressed.png
");
  Blynk.setProperty(V3, "url", "https://docs.blynk.io/en/getting-
started/what-do-i-need-to-blynk/how-quickstart-device-was-made");
}

// This function sends Arduino's uptime every second to Virtual Pin 2.
void myTimerEvent()
{
  // You can send any value at any time.
  // Please don't send more that 10 values per second.
  Blynk.virtualWrite(V2, millis() / 1000);

}

void setup()
{
  // Debug console
  Serial.begin(115200);

  Blynk.begin(auth, ssid, pass);
  // You can also specify server:
  //Blynk.begin(auth, ssid, pass, "blynk.cloud", 80);
  //Blynk.begin(auth, ssid, pass, IPAddress(192,168,1,100), 8080);

  // Setup a function to be called every second
  timer.setInterval(1000L, myTimerEvent);
  Serial.begin(9600);

  pinMode(ledPin, OUTPUT);
  pinMode(IR1ledPin, OUTPUT); // declare LED as output
  pinMode(IR1inputPin, INPUT); // declare Infrared sensor as input
  pinMode(IR2ledPin, OUTPUT); // declare LED as output
  pinMode(IR2inputPin, INPUT); // declare Infrared sensor as input
  pinMode(ldrPin, INPUT);
```

```
  attachInterrupt(digitalPinToInterrupt(IR1inputPin), detectsMovement1,
RISING);
  attachInterrupt(digitalPinToInterrupt(IR2inputPin), detectsMovement2,
RISING);
}
ICACHE_RAM_ATTR void detectsMovement1() {
  if(value==1){
    ldrStatus = analogRead(ldrPin);
    if (ldrStatus <=1000){
      Serial.println("MOTION DETECTED1!!!");
      digitalWrite(IR1ledPin, HIGH);
      startTimer1 = true;
      lastTrigger1 = millis();
      }
   }
}
ICACHE_RAM_ATTR void detectsMovement2() {
  if(value==1){
    ldrStatus = analogRead(ldrPin);
    if (ldrStatus <=1000){
      Serial.println("MOTION DETECTED2!!!");
      digitalWrite(IR2ledPin, HIGH);
      startTimer2 = true;
      lastTrigger2 = millis();
      }
   }
}
//void lrc1cont(){
//  if(lrc1==1){
//    ldrStatus = analogRead(ldrPin);
//    if (ldrStatus <=1000){
//      digitalWrite(lrc1_led, HIGH);
//      }
//    else{
//      digitalWrite(lrc1_led, LOW);
//    }
//   }
//   else{
//      digitalWrite(lrc1_led, LOW);
//    }
//}
//void lrc2cont(){
//  if(lrc2==1){
//    ldrStatus = analogRead(ldrPin);
//    if (ldrStatus <=1000){
//      digitalWrite(lrc2_led, HIGH);
//      }
//    else{
//      digitalWrite(lrc2_led, LOW);
//    }
//   }
//   else{
//      digitalWrite(lrc2_led, LOW);
//    }
```

```
//}
void loop()
{
  Blynk.run();
  timer.run();
  // You can inject your own code or combine it with other sketches.
  // Check other examples on how to communicate with Blynk. Remember
  // to avoid delay() function!
  ldrStatus = analogRead(ldrPin);
  Blynk.virtualWrite(V1, ldrStatus);
//  bridge1.virtualWrite(V0,value);
//  bridge1.virtualWrite(V1,ldrStatus);
//  lrc1cont();
//  lrc2cont();
  if(value==1){
    if (ldrStatus <=1000){
      digitalWrite(ledPin, HIGH);
      Serial.print(ldrStatus);
      Serial.println("LDR is DARK, LED is ON");
      now = millis();
      if(startTimer1&&((now-lastTrigger1)>3000)){
        digitalWrite(IR1ledPin, LOW);
        startTimer1 = false;
      }
      now = millis();
      if(startTimer2&&((now-lastTrigger2)>3000)){
        digitalWrite(IR2ledPin, LOW);
        startTimer2 = false;
      }
    }
    else{
    digitalWrite(ledPin, LOW);
    Serial.println("LED is OFF");
    digitalWrite(IR1ledPin, LOW);
    digitalWrite(IR2ledPin, LOW);
    }
   }
   else{
    digitalWrite(ledPin, LOW);
    digitalWrite(IR1ledPin, LOW);
    digitalWrite(IR2ledPin, LOW);
   }
}
```

```
/*************************************************************
  RUN THIS CODE ON NODE MCU NUMBER 05
  This is a simple demo of sending and receiving some data.
  Be sure to check out other examples!
 *************************************************************/

// Template ID, Device Name and Auth Token are provided by the Blynk.Cloud
// See the Device Info tab, or Template settings
#define BLYNK_TEMPLATE_ID "TMPLEzG0Eco2"
#define BLYNK_DEVICE_NAME "Project IOT MCU 5"
#define BLYNK_AUTH_TOKEN "FnEqTrh8QH8bsLhTJYTbQGJQO7U4CtHS"

// Comment this out to disable prints and save space
#define BLYNK_PRINT Serial


#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

char auth[] = BLYNK_AUTH_TOKEN;

// Your WiFi credentials.
// Set password to "" for open networks.
//char ssid[] = "Baheti";
//char pass[] = "P@SSW0RD123";
char ssid[] = "me.baheti.07";
char pass[] = "123456789";
int PUMP= 2;
int moisture = A0; // sensor input at Analog pin A0
int mois_value = 0;
int sys = 0;

//FOuntain System
int fnt_sys = 0;
const int IR1ledPin = 12; // choose pin for the LED
const int IR1inputPin = 13; // choose input pin (for Infrared sensor)
const int IR2ledPin = 14;
const int IR2inputPin = 15;
int IR1val = 0; // variable for reading the pin status
int IR2val = 0;

// Timer: Auxiliary variables
unsigned long now = millis();
unsigned long lastTrigger1 = 0;
boolean startTimer1 = false;

unsigned long lastTrigger2 = 0;
boolean startTimer2 = false;
```

```
BlynkTimer timer;

// This function is called every time the Virtual Pin 0 state changes
BLYNK_WRITE(V2)
{
  // Set incoming value from pin V1 to a variable
  fnt_sys = param.asInt();
}
// This function is called every time the device is connected to the
Blynk.Cloud
BLYNK_CONNECTED()
{
  // Change Web Link Button message to "Congratulations!"
  Blynk.setProperty(V3, "offImageUrl", "https://static-
image.nyc3.cdn.digitaloceanspaces.com/general/fte/congratulations.png");
  Blynk.setProperty(V3, "onImageUrl",  "https://static-
image.nyc3.cdn.digitaloceanspaces.com/general/fte/congratulations_pressed.png
");
  Blynk.setProperty(V3, "url", "https://docs.blynk.io/en/getting-
started/what-do-i-need-to-blynk/how-quickstart-device-was-made");
}

// This function sends Arduino's uptime every second to Virtual Pin 2.
void myTimerEvent()
{
  // You can send any value at any time.
  // Please don't send more that 10 values per second.
  Blynk.virtualWrite(V3, millis() / 1000);
}

void setup()
{
  // Debug console
  Serial.begin(19200);

  Blynk.begin(auth, ssid, pass);
  // You can also specify server:
  //Blynk.begin(auth, ssid, pass, "blynk.cloud", 80);
  //Blynk.begin(auth, ssid, pass, IPAddress(192,168,1,100), 8080);

  // Setup a function to be called every second
  timer.setInterval(1000L, myTimerEvent);
  pinMode(PUMP, OUTPUT); // declare pump relay as output
  pinMode(moisture, INPUT); // declare Infrared sensor as input
  pinMode(IR1ledPin, OUTPUT); // declare LED as output
  pinMode(IR1inputPin, INPUT); // declare Infrared sensor as input
  pinMode(IR2ledPin, OUTPUT); // declare LED as output
  pinMode(IR2inputPin, INPUT); // declare Infrared sensor as input
```

```
  attachInterrupt(digitalPinToInterrupt(IR1inputPin), detectsMovement1,
RISING);
  attachInterrupt(digitalPinToInterrupt(IR2inputPin), detectsMovement2,
RISING);
}
ICACHE_RAM_ATTR void detectsMovement1() {
  if(fnt_sys==1){
      Serial.println("MOTION DETECTED1!!!");
      digitalWrite(IR1ledPin, HIGH);
      startTimer1 = true;
      lastTrigger1 = millis();

   }
}
ICACHE_RAM_ATTR void detectsMovement2() {
  if(fnt_sys==1){
      Serial.println("MOTION DETECTED2!!!");
      digitalWrite(IR2ledPin, HIGH);
      startTimer2 = true;
      lastTrigger2 = millis();

   }
}
void loop()
{
  Blynk.run();
  timer.run();
  // You can inject your own code or combine it with other sketches.
  // Check other examples on how to communicate with Blynk. Remember
  // to avoid delay() function!
  mois_value = analogRead(A0);
  mois_value = map(mois_value, 0, 1023, 0, 100);
  Blynk.virtualWrite(V5, mois_value);
  if(fnt_sys==1){
      now = millis();
      if(startTimer1&&((now-lastTrigger1)>3000)){
        digitalWrite(IR1ledPin, LOW);
        startTimer1 = false;
      }
      now = millis();
      if(startTimer2&&((now-lastTrigger2)>3000)){
        digitalWrite(IR2ledPin, LOW);
        startTimer2 = false;
      }
   }
   else{
    digitalWrite(IR1ledPin, LOW);
    digitalWrite(IR2ledPin, LOW);
```

```
   }
  Serial.println(sys);
    if (mois_value > 50 ){
        digitalWrite(PUMP, HIGH); //
    }
    else{
    Serial.println("PUMP is OFF");
    digitalWrite(PUMP, LOW);
    }
}
```