College of Engineering

PBL REPORT

In Partial Fulfillment of the Requirements for

ECE 415 - Microprocessor and Microcontroller Systems and Design

Bachelor of Science in Instrumentation and Control Engineering

Submitted to:

Engr. Gil Barte

Submitted by:

Arellano, Kevin B.

Casalla, Joaquin Angelo

Mendaña, Sherine E.

Project Title:

No-Man, No-Power

I. Introduction

With the responsibilities of attending classes and doing coursework, safe energy management becomes an afterthought for students living independently, as does maintaining safety at home. It's fairly common for students to forget to turn off electric fans, rice cookers, or lights while leaving a room. Such oversight not only escalates electricity wastage but also increases utility expenditures and safety risks like fire and other hazards.

By creating an automated system that monitors room occupancy, the No-One, No-Power project alleviates these concerns. Such a system design: with the simple technology of a motion sensor and microcontroller, it can automatically disable connected appliances after a reasonable time of no activity in a monitored room. The system aims to enhance energy utilization, save costs, and improve safety for students and others living alone.

II. Objectives

General Objective

To design and develop an automated system that cuts off the power supply to appliances after a period of inactivity, helping students and individuals living alone save electricity, reduce utility costs, and improve safety.

Specific Objectives

a. To create a motion detection system using a PIR sensor that continuously monitors the presence of people in a room.

b. To implement a timer mechanism that starts counting when no motion is detected.

c. To integrate a relay control module that automatically disconnects power supply when the timer reaches 30 minutes of inactivity.

d. To ensure that the system resets the timer immediately if any movement is detected within the countdown period.

e. To develop a safe and reliable circuit that can handle typical household appliance loads.

f. To promote energy conservation by minimizing unnecessary power consumption.

III. Consideration

Hardware Considerations:

- **Component Class:** Include rated and reliable components such as the relay and AC power wiring.

- **Appropriate Construction:** Tighten and insulate every connection on the circuit lest they short and lead to faults.

- **Sensor Location:** Install the PRS sensor in an area where it can cover most space and have the least amount of blind spots.

- **Stability of the Power Supply:** Make sure that there is constant supplied power to eliminate chances of failure and random resets.

- **Device Protections:** Protect the circuit from dust, moisture, and inadvertent threats by putting the device in a casing.

Software Considerations:

- **Timer Task:** Set inactivity counters under no motion conditions to ensure no drift occurs.

- **Fail-Safe Behavior:** Ensure that sensor inputs are processed swiftly to eliminate chances of false triggering for action recognition.

- **Reset Mechanisms:** Upon performing the deem, reset actions.

Testing and Evaluation Considerations:

- **Accuracy:** Confirm precision and check actions, both detecting and undetecting, are correctly identified by the motion sensor.

- **Timer Functionality:** Ensure that the 30-minute timer resets properly upon detection of movement.

- **Relay Operation:** Confirm and check whether the relay is concerned with delays or glitches that indisputably cut off and restore power.

- **Environmental Conditions:** Assess performance under different levels of lighting/temperature and room arrangements.

- **Safety Assessments:** Conduct evaluation to confirm operation does not present electrical risks.

IV. Design Constraint

a. **Timeframe:** We have a limited time to complete the project

V. Working Principles

1. With a usual installation in the room, PIR motion detectors are the ones that start the system when movement of human beings generates infrared heat.

2. The detection sensors will pass on the detection information to the microcontroller, which could either be an Arduino or an ESP32.

3. The microcontroller receives the information from the sensor and processes it to control the relay module for powering on home appliances like lights and fans.

4. If no movement is detected for a certain time (30 minutes), it assumes that there is no one in the room and gives an instruction to the relay not to supply power to those appliances.

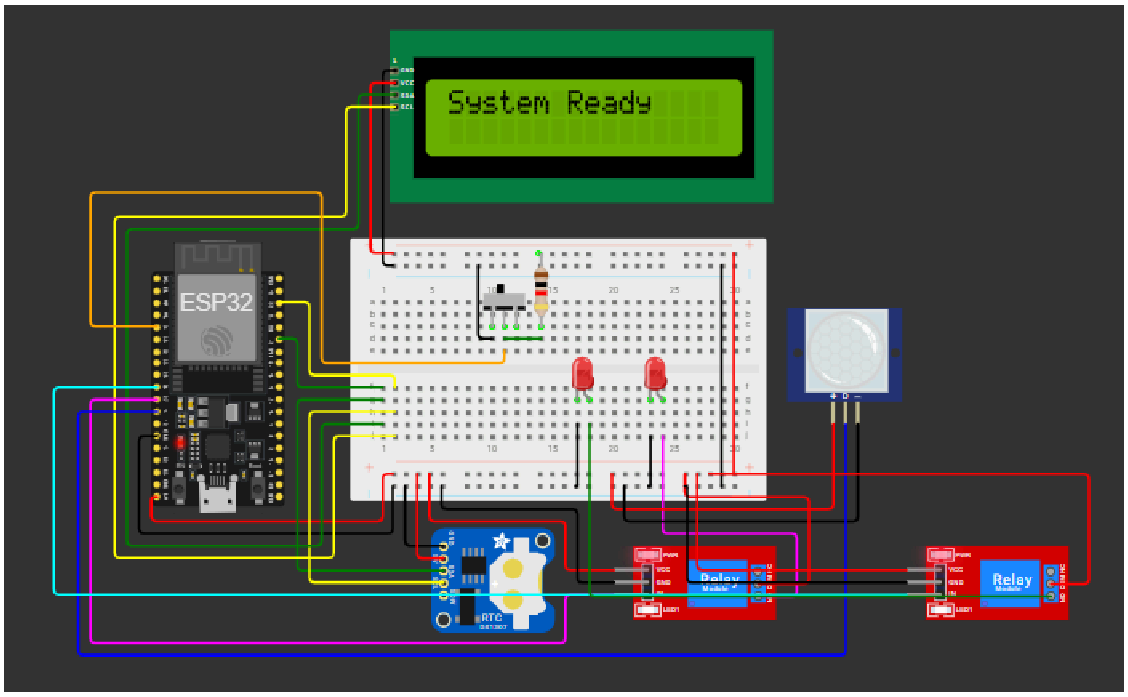
5. Upon detection of motion again, the microcontroller is triggered through the sensor to restart the process and re-activate the appliances.

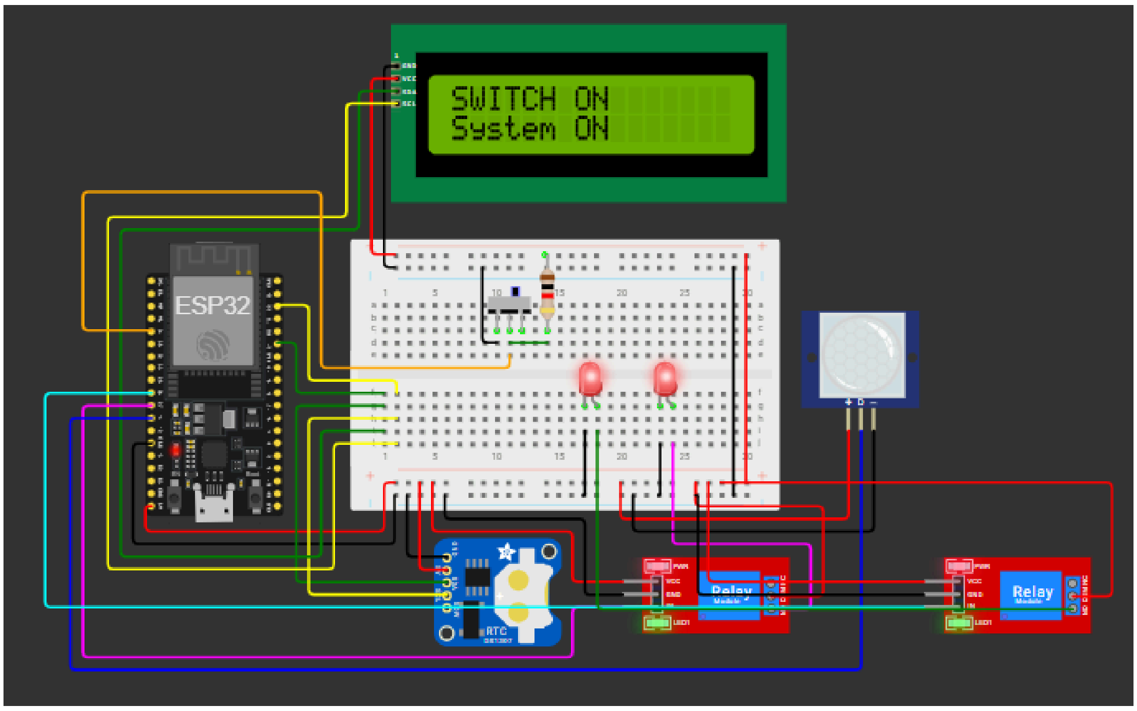
6. A power supply ensures that all sensor components (sensor, microcontroller, and relay) are kept under a constant power supply to give them consistent voltages.

VI. Essential Components

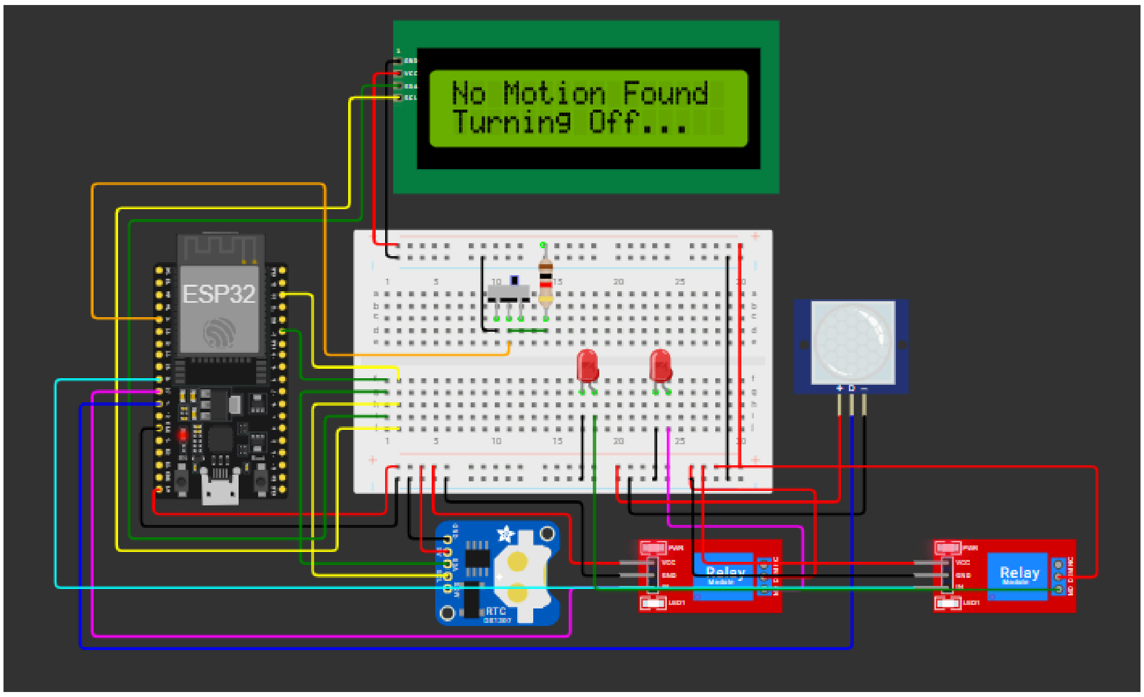
1. **ESP32:**
   - The main microcontroller that controls all the components. It reads the inputs from the switch and PIR sensor and controls the relay and servo motor based on the program.
2. **PIR Sensor:**
   - Detects motion in the environment. When motion is detected, it signals the ESP32 to turn on the light (through the relay).
3. **Relay Module:**
   - Acts as an electronic switch that controls a separate high-power circuit using the ESP32's low-power signal.
4. **RTC Module:**
   - Keeps track of the real-world time even if the ESP32 loses power. This is used to allow the system to work only within a set time range.
5. **LED and DC Motors (*Standard Appliances)***
   - Represents a load controlled by the relay. It turns ON or OFF based on motion detection.
6. **Switch**
   - Acts as a master ON/OFF switch for the whole system. When the switch is OFF, all functions are disabled. When the switch is ON, the system starts detecting motion and controlling the LED and servo motor.

VII. Hardware Schematics

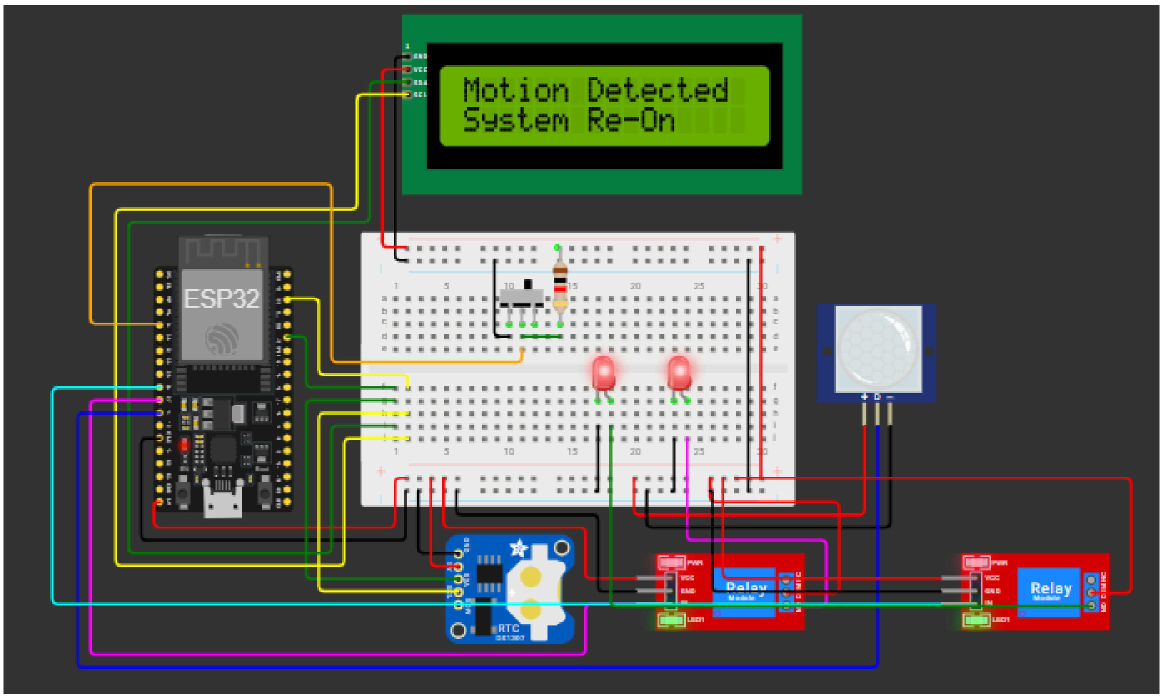




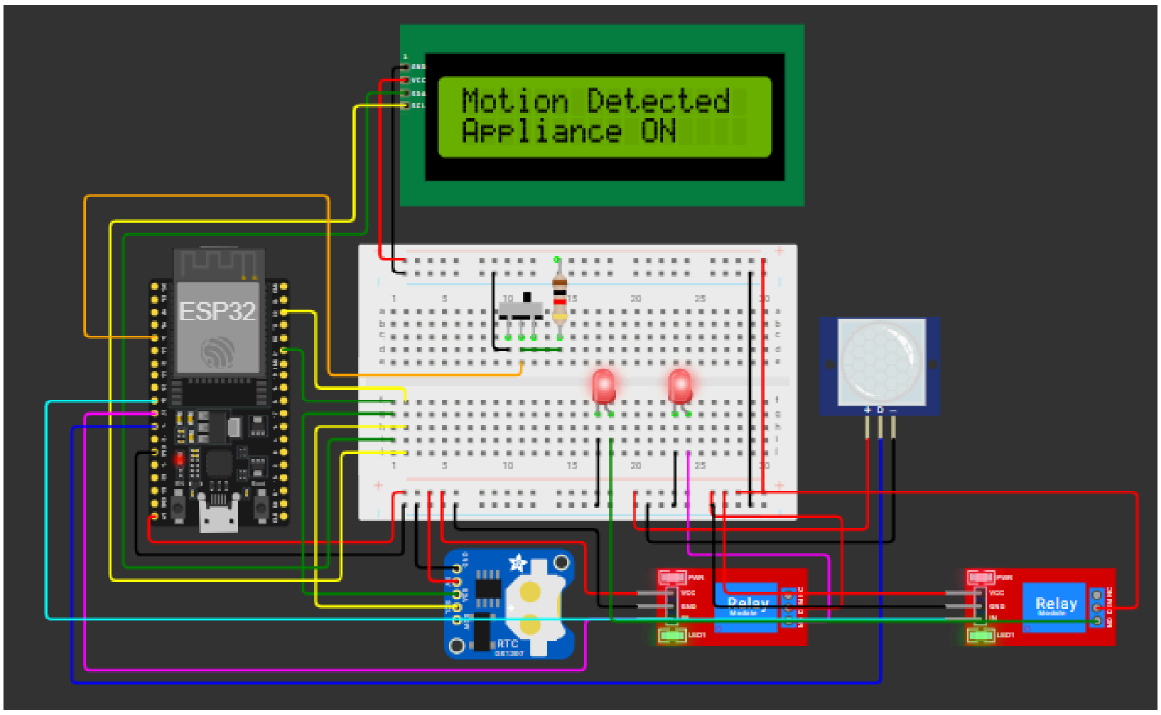
```
SWITCH ON
System ON
```



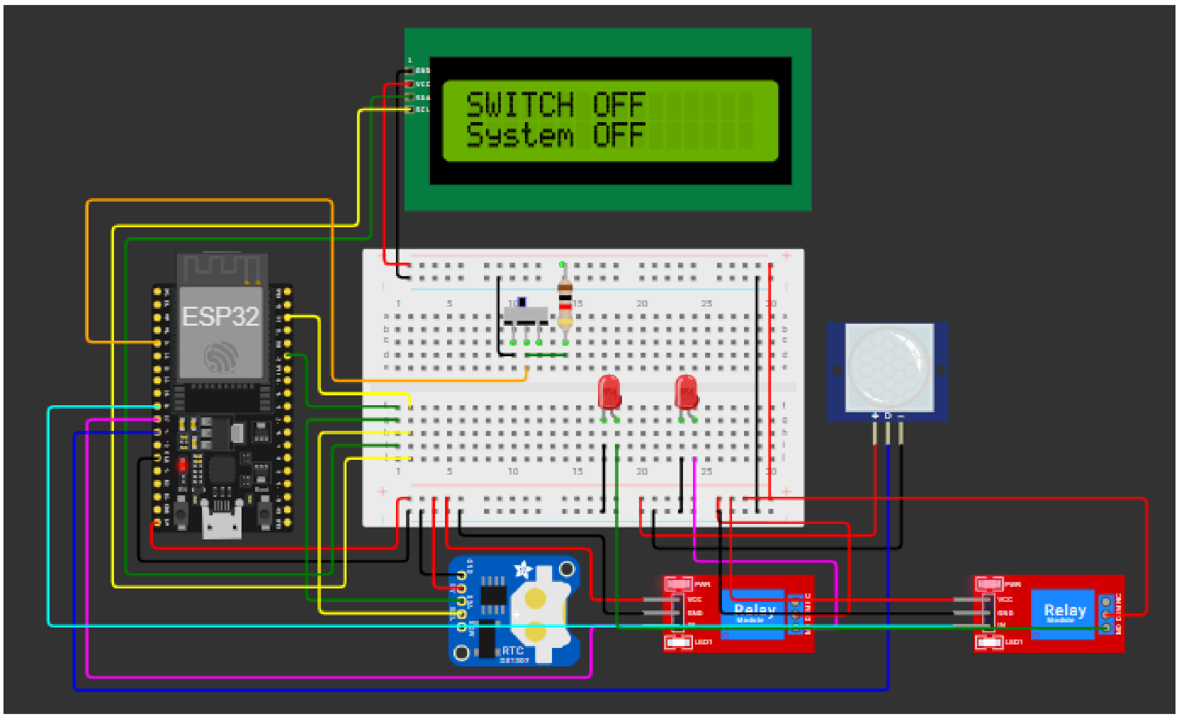
```
No Motion Found
Turning Off...
```



```
Motion Detected
System Re-On
```



Motion Detected
Appliance ON



SWITCH OFF
System OFF

VIII. Code

```
#define BLYNK_TEMPLATE_ID “TMPL61XBA8oxd”

#define BLYNL_TEMPLETE_NAME “No Man No Power”

#define BLYNK_AUTH_TOKEN “Zke3cE1nF01mW3J9-4fw5XGqCjn90SJQ”



#include <Wifi.h>

#include <WifiClient.h>

#include <BlynkSimpleEsp32.h>

#include <Wire.h>

#include <RTClib.h>
```

```
#include <LiquidCrystal_12C.h>

RTC_DS3231 rtc;
LiquidCrystak_12C lcd(0x27, 16, 2)

#define PIR_PIN 14
#define RELAY1_PIN 27
#define RELAY2_PIN 26
#define SWITCH_PIN 34
#define BLYNK_ONOFF_VPIN V0

char sside[] = “YourWifiSSID”;
char pass[] = “YourWifiPassword”

unsigned long lastMotionTIme = 0
unsigned long motionTImeoute = 30000
bool systemOn = false;
bool systemSleeping = false;
bool blynkEnabled = true;

BLYNK_WRITE(BLYNK_ONOFF_VPIN) {
    blynkEnabled = param.asInt();
}
void setup() {
Serial.begin(115200);
pinMode(PIR_PIN, INPUT);
pinMode(RELAY1_PIN, OUTPUT);
pinMode(RELAY2_PIN, OUTPUT);
pinMode(SWITCH_PIN, INPUT);

digitalWrite(RELAY1_PIN, LOW);
digitalWrite(RELAY2_PIN, LOW);
```

```
Wire.begin();

rtc.begin();

lcs.init();

lcd.backlight();

lcd.setCursor(0, 0);

lcd.print(“Connecting WIFI”);


Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print(“System Ready”);

}

void loop() {

Blynk.run();


bool switchState = digitalRead(SWITCH_PIN) == HIGH;


if (!switchstate // !blynkEnabled) {

  if (systemOn // systemSleeping) {

    lcd.clear():

    lcd.setCursor(0, 0);

    lcd.print(“switchState ? “Blynk: OFF” : SWITCH: OFF”);

    lcd.setCursor(0, 1);

    lcd.print(“System OFF)”

    digitalWrite(RELAY1_PIN, LOW);

    digitalWrite(RELAY2_PIN, LOW);

    systemOn = false;

    systemSleeping = false;

}

delay(500);

return;

}
```

```
if (!systemon && !systemSleeping) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(“SWITCH ON”);
    lcd.setCursor(0, 1);
    lcd.print(“System ON”);
    digitalWrite(RELAY1_PIN, HIGH);
    digitalWrite(RELAY2_PIN, HIGH);
    systemOn = true;
    lastMotionTime = millis();
    delay(1000);
}

if (systemOn && digitalRead(PIR_PIN) == HIGH) {
      lastMotionTime = millis();
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print(“Motion Detected”);
      lcd.setCursor(0, 1);
      lcd.print(“Appliance ON”);
      digitalWrite(RELAY1_PIN, HIGH);
      digitalWrite(RELAY2_PIN, HIGH);
}

if (systemOn && (millis() - lastMotionTime > motionTimeout)) {
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print(“No Motion Found”);
      lcd.setCursor(0, 1);
      lcd.print(“Turning OFF…”);
      digitalWrite(RELAY1_PIN, LOW);
      digitalWrite(RELAY2_PIN, LOW);
    systemOn = false;

    systemSleeping = true;

    delay (2000);

}

if (systemSleeping && digitalRead(PIR_PIN) == HIGH) {

    lcd.clear();

    lcd.setCursor(0, 0);

    lcd.print(“Motion Detected”);

    lcd.setCursor(0, 1);

    lcd.print(“System Re-On”);

    digitalWrite(RELAY1_PIN, HIGH);

    digitalWrite(RELAY2_PIN, HIGH);

    systemOn = true;

    systemSleeping = false;

    delay (2000);

}

delay (200);

}
```

IX. Pseudocode

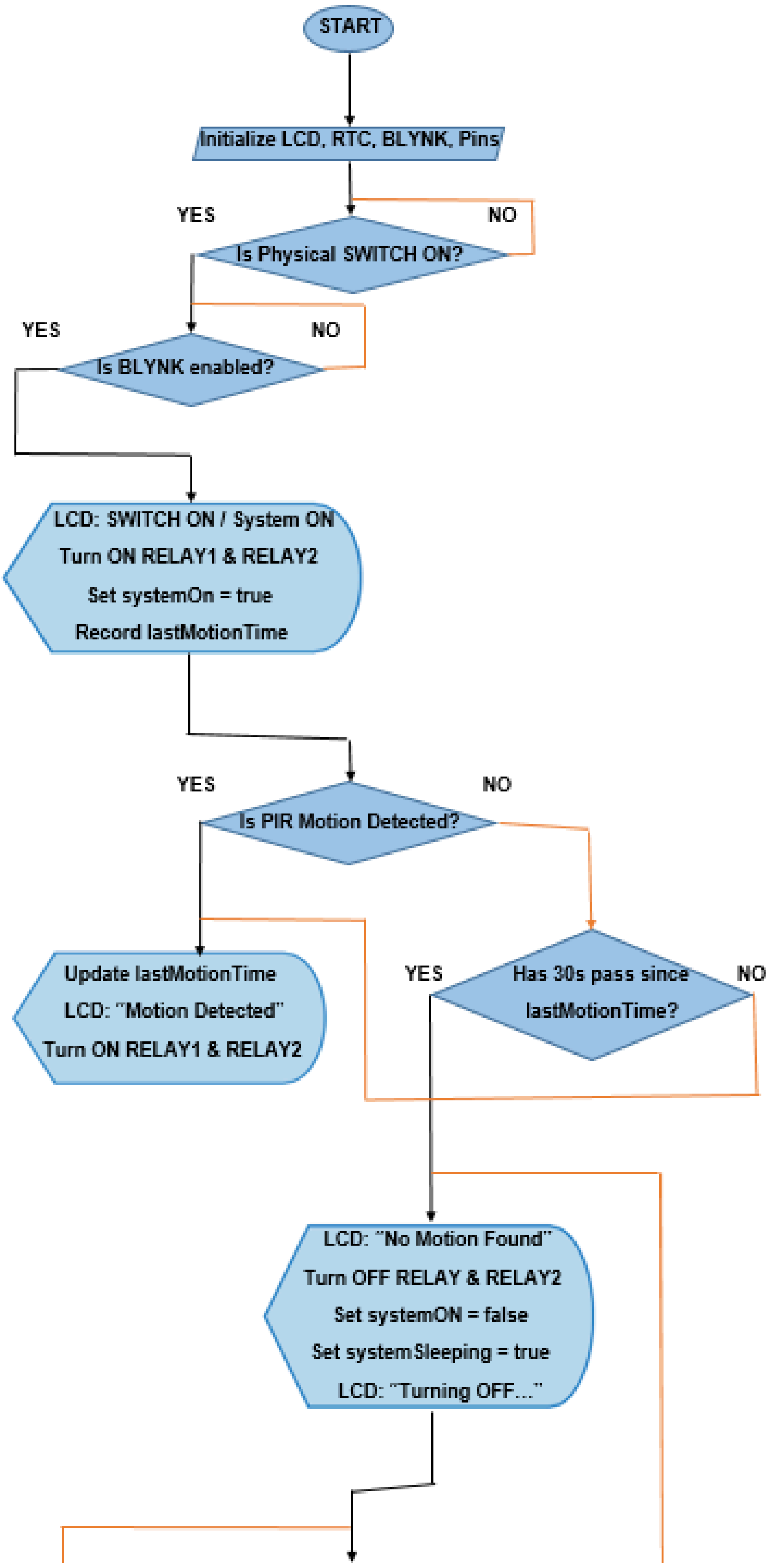
```
1. IDEFINE Blynk template credentials
2. Include necessary libraries (WiFi, Blynk, RTC, LCD)
3. INITIALIZE:
       -  RTC module
       -  LCD with 12C
       -  Pins for PIR sensor, relays, physical switch
4. SET system state variables:
       -  lastMotionTime = 0
       -  motionTimeout = 30 seconds
       -  systemOn = false
       -  systemSleeping = false
       -  blynkEnabled = true
5. CONNECT to WiFi and Blynk
6. DISPLAY “System ready” on LCD

   Loop forever:

7. RUN Blynk background processes
8. READ physical switch state
```

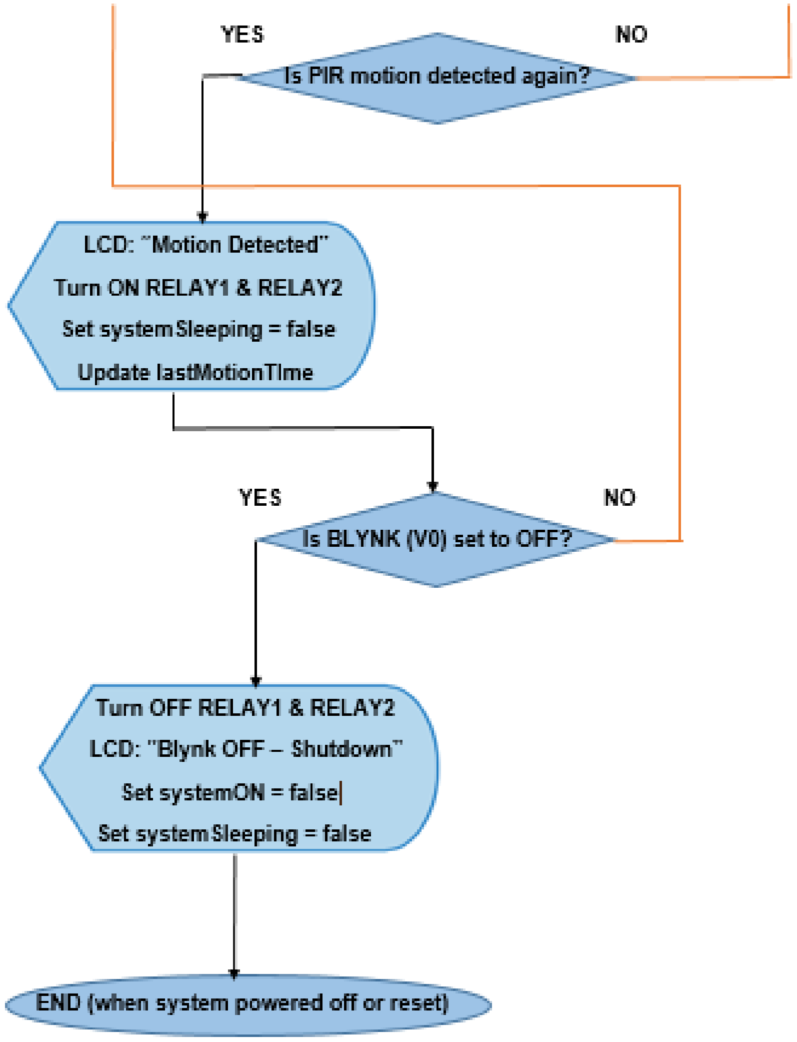
```
9.  IF (switch is OFF OR Blynk is OFF):
        -   DISPLAY status on LCD
        -   TURN OFF all relays
        -   RESET systemOn and systemSleeping flags
        -   WAIT briefly
        -   SKIP rest of loop (return)
10. IF (system is not ON and not sleeping):
        -   DISPLAY “System ON” on LCD
        -   TURN ON relays (motor and LED)
        -   SET systemOn = true
        -   RECORD current time as lastMotionTime
11. IF (motion is detected);
        -   UPDATE lastMotionTime
        -   DISPLAY “Motion Detected”
        -   ENSURE relays stay ON
12. OIF (no motion for 30 seconds);
        -   DISPLAY “Turning Off…” on LCD
        -   TURN OFF relays
        -   SET systemOn = false
        -   SET systemSleeping = true
13. IF (system is sleeping AND motion is detected again):
        -   DISPLAY “System Re-On”
        -   TURN ON relays
        -   RESET systemSleeping = false
        -   SET systemOn = true
        -   UPDATE lastMotionTIme
14. WAIT 200 ms before repeating

    END LOOP
```

X. Downloaded Necessary Library

- **Wire**
- **RTClib**
- **WiFi**
- **Blynk**
- **LCD**

XI. Flow Chart




YES
NO
Is PIR motion detected again?
LCD: "Motion Detected"
Turn ON RELAY1 & RELAY2
Set systemSleeping = false
Update lastMotionTIme
YES
NO
Is BLYNK (V0) set to OFF?
Turn OFF RELAY1 & RELAY2
LCD: "Blynk OFF – Shutdown"
Set systemON = false|
Set systemSleeping = false
END (when system powered off or reset)

XII. Contributions of the Members

- **Arellano, Kevin B.**
    - Essential Components to use
    - Constructing the Hardware Circuit and on Wokwi
    - Wokwi and Hardware Circuit Simulation
    - Program Code
    - Pseudocode
    - Flowchart
    - Making the PBL work
    - BLYNK IoT Configuration
    - Paper
- **Mendana, Sherine**
    - Parts of the paper from Chapter 1-5

XIII. Wokwi Project Link

https://wokwi.com/projects/430552444595402753