Software Documentation

Project Title: Indicator Control Over Mobile Phones (Using BLE Protocol)

Author: Hemanth Kumar J N

---

1. Introduction

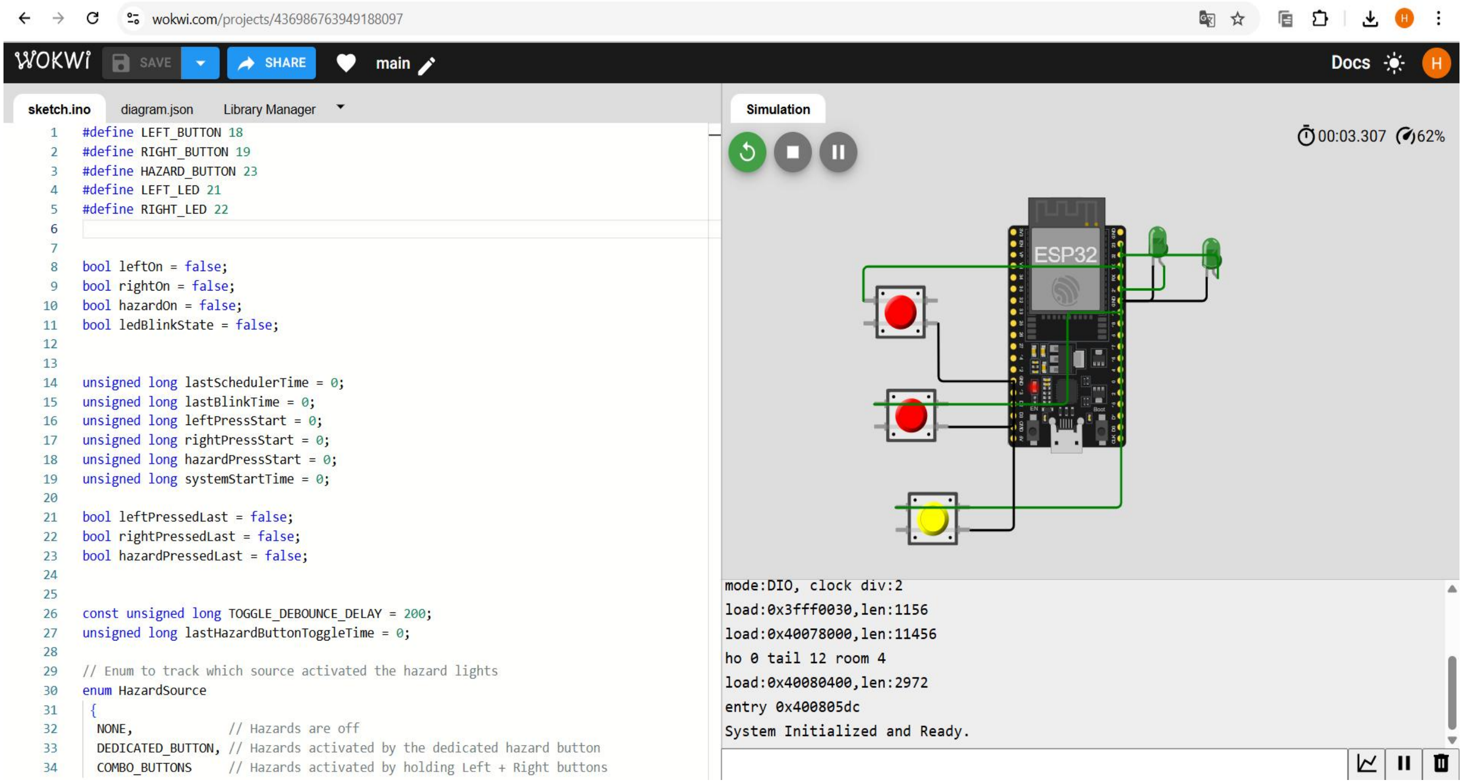
This document presents the design and implementation of a simulated vehicle indicator system, developed using an ESP32 microcontroller within the Wokwi simulation platform. Due to limitations in accessing physical hardware, all functionalities were verified virtually. The system includes left, right, and hazard indicators, controlled via GPIO-based push buttons and provides UART feedback for logging and debugging.

---

2. High-Level Architecture

2.1 Hardware Architecture (Simulated)

Component	Function
ESP32 DevKit v4	Main microcontroller
3 Push Buttons	Left, Right, and Hazard control inputs
2 LEDs	Left and Right indicator output signals
UART Interface	Serial monitor for debugging/logging



```
#define LEFT_BUTTON 18
#define RIGHT_BUTTON 19
#define HAZARD_BUTTON 23
#define LEFT_LED 21
#define RIGHT_LED 22


bool leftOn = false;
bool rightOn = false;
bool hazardOn = false;
bool ledBlinkState = false;


unsigned long lastSchedulerTime = 0;
unsigned long lastBlinkTime = 0;
unsigned long leftPressStart = 0;
unsigned long rightPressStart = 0;
unsigned long hazardPressStart = 0;
unsigned long systemStartTime = 0;

bool leftPressedLast = false;
bool rightPressedLast = false;
bool hazardPressedLast = false;


const unsigned long TOGGLE_DEBOUNCE_DELAY = 200;
unsigned long lastHazardButtonToggleTime = 0;

// Enum to track which source activated the hazard lights
enum HazardSource
 {
  NONE,             // Hazards are off
  DEDICATED_BUTTON, // Hazards activated by the dedicated hazard button
  COMBO_BUTTONS     // Hazards activated by holding Left + Right buttons
```

```
mode:DIO, clock div:2
load:0x3fff0030,len:1156
load:0x40078000,len:11456
ho 0 tail 12 room 4
load:0x40080400,len:2972
entry 0x400805dc
System Initialized and Ready.
```

fig(1): *Wokwi setup, showing button and LED wiring)*

2.2 Software Architecture

- **GPIO Control:**
  - INPUT_PULLUP for button inputs (active LOW)
  - OUTPUT for LED control
- **Scheduler:**
  - 100ms task scheduler for button polling
  - 300ms toggle timer for LED blinking
- **UART Logging:**
  - Real-time UART logs sent over serial monitor
- **Hazard Control:**
  - Activated either by holding both Left and Right buttons for 1s or via a dedicated button

3. Functional Requirements

Feature	Description
Indicator Toggle	Press and hold Left or Right button for 1 second to toggle respective indicator
Exclusive Mode	Only one indicator can be ON at a time
Hazard Mode via Combo	Press and hold both buttons for >1s to activate hazard mode
Hazard via Dedicated Button	Press hazard button to toggle hazard state
LED Blinking	Active indicator LEDs toggle ON/OFF every 300ms
UART Logging	Status messages printed: Button events, state changes, activations

4. Implementation Overview

Development Environment:

- Platform: Wokwi ESP32 Simulator
- Language: C/C++ (Arduino framework)
- Core Libraries Used: Arduino.h, Serial, GPIO functions

Working Description:

- On boot, the system initializes GPIO and timers.
- Every 100ms, button states are read and processed.
- Holding buttons for more than 1 second determines toggling or hazard activation.
- When any indicator is active, its LED toggles ON/OFF every 300ms.
- UART logs report status in real-time for debugging and monitoring.

5. Simulation Results

- LED toggle timing verified using Serial log timestamps
- Button presses reflected correctly in console output
- Hazard logic validated by both methods
- Reliable debounce and edge detection simulated correctly

UART Log Sample:

```
System Initialized and Ready.
Left button was just pressed.
Left Indicator ON.
Right button was just pressed.
Left Indicator OFF.
Right Indicator ON.
Hazard Lights ACTIVATED by Dedicated Button.
Hazard Lights DEACTIVATED by Dedicated Button.
```

---

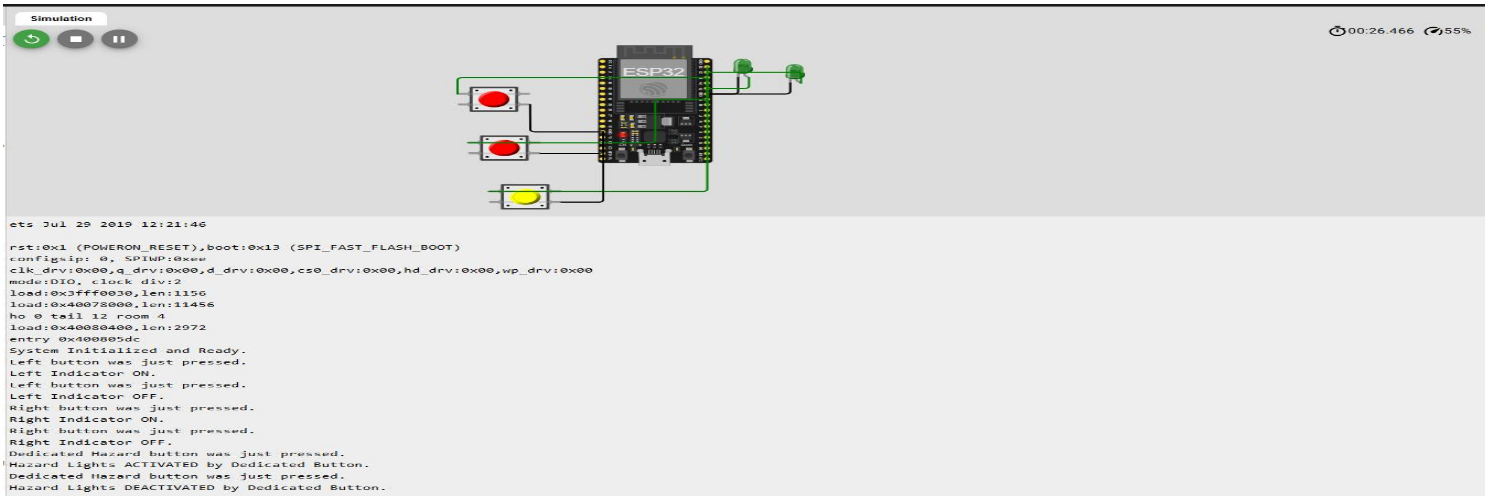
6. Code Snippet (.ino)

```
#define LEFT_BUTTON 18
#define RIGHT_BUTTON 19
#define HAZARD_BUTTON 23
#define LEFT_LED 21
#define RIGHT_LED 22

bool leftOn = false;
bool rightOn = false;
bool hazardOn = false;
bool ledBlinkState = false;

unsigned long lastSchedulerTime = 0;
unsigned long lastBlinkTime = 0;
unsigned long leftPressStart = 0;
unsigned long rightPressStart = 0;
unsigned long hazardPressStart = 0;
unsigned long systemStartTime = 0;

bool leftPressedLast = false;
bool rightPressedLast = false;
bool hazardPressedLast = false;

const unsigned long TOGGLE_DEBOUNCE_DELAY = 200;
unsigned long lastHazardButtonToggleTime = 0;

enum HazardSource {
  NONE,
  DEDICATED_BUTTON,
  COMBO_BUTTONS
};
HazardSource currentHazardSource = NONE;

void setup() {
  pinMode(LEFT_BUTTON, INPUT_PULLUP);
  pinMode(RIGHT_BUTTON, INPUT_PULLUP);
  pinMode(HAZARD_BUTTON, INPUT_PULLUP);
  pinMode(LEFT_LED, OUTPUT);
  pinMode(RIGHT_LED, OUTPUT);
  digitalWrite(LEFT_LED, LOW);
  digitalWrite(RIGHT_LED, LOW);
  Serial.begin(115200);
  delay(500);
  leftPressedLast = digitalRead(LEFT_BUTTON) == LOW;
  rightPressedLast = digitalRead(RIGHT_BUTTON) == LOW;
  hazardPressedLast = digitalRead(HAZARD_BUTTON) == LOW;
  systemStartTime = millis();
  leftPressStart = leftPressedLast ? systemStartTime : 0;
  rightPressStart = rightPressedLast ? systemStartTime : 0;
  hazardPressStart = hazardPressedLast ? systemStartTime : 0;
  Serial.println("System Initialized and Ready.");
}

void updateLEDs() {
  int ledVal = ledBlinkState ? HIGH : LOW;
  if (hazardOn) {
    digitalWrite(LEFT_LED, ledVal);
    digitalWrite(RIGHT_LED, ledVal);
  } else if (leftOn) {
    digitalWrite(LEFT_LED, ledVal);
    digitalWrite(RIGHT_LED, LOW);
  } else if (rightOn) {
    digitalWrite(RIGHT_LED, ledVal);
    digitalWrite(LEFT_LED, LOW);
  } else {
    digitalWrite(LEFT_LED, LOW);
    digitalWrite(RIGHT_LED, LOW);
  }
}

void handleButtons(unsigned long currentTime) {
  if (currentTime - systemStartTime < 1000) {
    leftPressedLast = digitalRead(LEFT_BUTTON) == LOW;
    rightPressedLast = digitalRead(RIGHT_BUTTON) == LOW;
    hazardPressedLast = digitalRead(HAZARD_BUTTON) == LOW;
    return;
  }
  bool leftPressedNow = digitalRead(LEFT_BUTTON) == LOW;
  bool rightPressedNow = digitalRead(RIGHT_BUTTON) == LOW;
  bool hazardPressedNow = digitalRead(HAZARD_BUTTON) == LOW;
  if (leftPressedNow && !leftPressedLast) leftPressStart = currentTime;
  if (rightPressedNow && !rightPressedLast) rightPressStart = currentTime;
  if (hazardPressedNow && !hazardPressedLast) hazardPressStart = currentTime;
  if (!hazardPressedNow && hazardPressedLast) {
    if (currentTime - lastHazardButtonToggleTime > TOGGLE_DEBOUNCE_DELAY) {
      lastHazardButtonToggleTime = currentTime;
      hazardOn = !hazardOn;
      currentHazardSource = hazardOn ? DEDICATED_BUTTON : NONE;
      leftOn = false;
      rightOn = false;
    }
  }
  if (leftPressedNow && rightPressedNow &&
      (currentTime - leftPressStart > 1000) &&
      (currentTime - rightPressStart > 1000)) {
    if (!hazardOn || currentHazardSource == COMBO_BUTTONS) {
      hazardOn = true;
      currentHazardSource = COMBO_BUTTONS;
      leftOn = false;
      rightOn = false;
    }
  } else {
    if (hazardOn && currentHazardSource == COMBO_BUTTONS) {
      hazardOn = false;
      currentHazardSource = NONE;
    }
  }
  if (hazardOn) {
    leftPressedLast = leftPressedNow;
    rightPressedLast = rightPressedNow;
    hazardPressedLast = hazardPressedNow;
    return;
  }
  if (!leftPressedNow && leftPressedLast) {
    leftOn = !leftOn;
    rightOn = false;
  }
  if (!rightPressedNow && rightPressedLast) {
    rightOn = !rightOn;
    leftOn = false;
  }
  leftPressedLast = leftPressedNow;
  rightPressedLast = rightPressedNow;
  hazardPressedLast = hazardPressedNow;
}
```

```
void loop() {
  unsigned long currentTime = millis();
  if (currentTime - lastSchedulerTime >= 100) {
    lastSchedulerTime = currentTime;
    handleButtons(currentTime);
  }
  if (currentTime - lastBlinkTime >= 300) {
    lastBlinkTime = currentTime;
    ledBlinkState = !ledBlinkState;
    updateLEDs();
  }
}
```

7. Repository and Resources

- **GitHub Code Link:** https://github.com/hemanthkumarjn05/vehicle_indicator_control.git
- **Demo Video (Google Drive):**
- **UART Log File:**



```
ets Jul 29 2019 12:21:46

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0030,len:1156
load:0x40078000,len:11456
ho 0 tail 12 room 4
load:0x40080400,len:2972
entry 0x400805dc
System Initialized and Ready.
Left button was just pressed.
Left Indicator ON.
Left button was just pressed.
Left Indicator OFF.
Right button was just pressed.
Right Indicator ON.
Right button was just pressed.
Right Indicator OFF.
Dedicated Hazard button was just pressed.
Hazard Lights ACTIVATED by Dedicated Button.
Dedicated Hazard button was just pressed.
Hazard Lights DEACTIVATED by Dedicated Button.
```

8. Conclusion & Acknowledgment

Due to limited access to ESP32 hardware, the full system was developed and tested using the Wokwi simulator. All core functionalities meet the given assignment specifications, including hazard behavior, UART communication, and modular task design.

I sincerely request you to kindly consider this submission for evaluation. If I had access to the real hardware, I would have implemented and tested it physically. Please grant me the opportunity to demonstrate the real-time setup once the required hardware is available.

Thank you!

– Hemanth Kumar J N