A

PROJECT REPORT

On

REAL-TIME OCCUPANCY DETECTION

Prepared by

ARUN GEORGE VIJI

Email ID : arungeorge.viji2021@vitstudent.ac.in

GitHub Link : https://github.com/ArunViji/MC-PROJECT.git

Submitted in partial fulfilment of the requirement for the completion of Global Internship

on

Industrial Applications of Microcontrollers - A Practice Based Approach

1. PROBLEM STATEMENT

The problem at hand is to develop a simple prototype that mimics a real-time occupancy detection system, which ultimately offers energy efficiency. The goal is to automatically control room lights based on the presence of humans, thereby optimizing energy consumption.

Context and Significance:

In many environments, such as offices, classrooms, or homes, lights are often left turned on even when there is no one present in the room. This leads to unnecessary energy wastage and increased electricity bills. By implementing a real-time occupancy detection system, we can address this issue and promote energy efficiency.

The significance of this problem lies in its potential to reduce energy consumption and contribute to sustainability efforts. By automatically turning off lights when a room is unoccupied, we can minimize energy waste and lower carbon emissions. This solution can be particularly beneficial in large buildings or spaces where manual control of lights may not be practical or efficient.

Existing Solutions and Limitations:

There are already existing solutions and approaches to occupancy detection systems. Some common methods include using motion sensors, infrared sensors, or ultrasonic sensors to detect human presence. These sensors can trigger the lights to turn on or off based on the detected motion or occupancy.

However, these existing solutions may have certain limitations.

- Motion sensor based switches - Only detects motion and not occupancy. Lights may turn OFF when someone is still in the room but not moving.
- Infrared occupancy sensors - Can detect presence but not entry/exit of people. Requires manual configuration of detection range and field of view. May malfunction with sudden changes in temperature.
- Image processing based solutions - Require installation of cameras raising privacy concerns. High computational requirements and costs.

The proposed solution aims to address limitations of existing approaches by using a combination of ultrasonic sensors for counting and PIR sensor for motion detection to

accurately determine occupancy and automatically control room lighting accordingly to save energy.

2. SCOPE OF THE SOLUTION

The scope of the solution involves developing a prototype that mimics a real-time occupancy detection system to offer energy efficiency. The solution aims to automatically control room lights based on the presence of humans, optimizing energy consumption in various environments such as offices, classrooms, or homes.

Boundaries and Objectives:

The scope of this project is limited to developing a proof-of-concept prototype to automatically control room lighting based on occupancy detection. The prototype will be tested on a single room setup. The prototype will focus on accurately detecting the number of people entering and leaving the room using the ultrasonic sensors, while the PIR sensor will detect motion inside the room.

The objective is to develop a low-cost, energy efficient solution to replace manual switching of lights with an automated system using sensors. They include:

1. Automatically turning on the lights when a person enters the room.

2. Keeping the lights on as long as there is motion detected inside the room.

3. Turning off the lights when there is no motion detected and no one present in the room.

4. Implementing a delay mechanism to ensure the lights remain on for a certain period after motion ceases.

Constraints or Limitations:

While the proposed solution offers a cost-effective and energy-efficient approach, there may be certain constraints or limitations to consider. Some of these include:

1. Accuracy: The accuracy of occupancy detection may be influenced by factors such as sensor placement, sensitivity settings, and environmental conditions. It is important to calibrate and test the system to ensure reliable detection.

2. Sensor Range: The range of the ultrasonic sensors and PIR sensor may have limitations. It is crucial to select sensors with appropriate ranges to cover the desired area effectively.
3. False Triggers: The system may be susceptible to false triggers caused by factors like pets, moving objects, or sudden changes in temperature. Careful consideration should be given to minimize false triggers and ensure reliable operation.
4. Integration: Integrating the prototype with existing lighting systems or infrastructure may require additional considerations and compatibility checks.

3. COMPONENTS REQUIRED

a) Arduino UNO R3 (Quantity: 1)

Specifications:

- Microcontroller: ATmega328P
- Operating Voltage: 5V
- Digital I/O Pins: 14
- Analog Input Pins: 6
- Flash Memory: 32KB
- SRAM: 2KB
- Clock Speed: 16MHz

b) Ultrasonic Sensor HC-SR04 (Quantity: 2):

Specifications:

- Operating Voltage: 5V
- Detection Range: 2cm - 400cm
- Working Frequency: 40kHz

c) PIR Motion Sensor (Quantity: 1):

Specifications:

- Operating Voltage: 5V
- Detection Range: Up to 7m
- Detection Angle: 120°

d) Relay SPDT (Quantity: 1):

Specifications:

- Type: Single Pole Double Throw (SPDT)
- Operating Voltage: 5V
- Contact Rating: 10A at 125VAC, 10A at 30VDC

e) **Bulb (Quantity: 1):**

Specifications:

- Voltage: 12V
- Wattage: 60W

f) **Power Supply (Quantity: 1):**

Specifications:

- Input: 100-240V AC
- Output: 12V DC, 2A

g) **Breadboard:**

The breadboard is used for prototyping and connecting the components together.

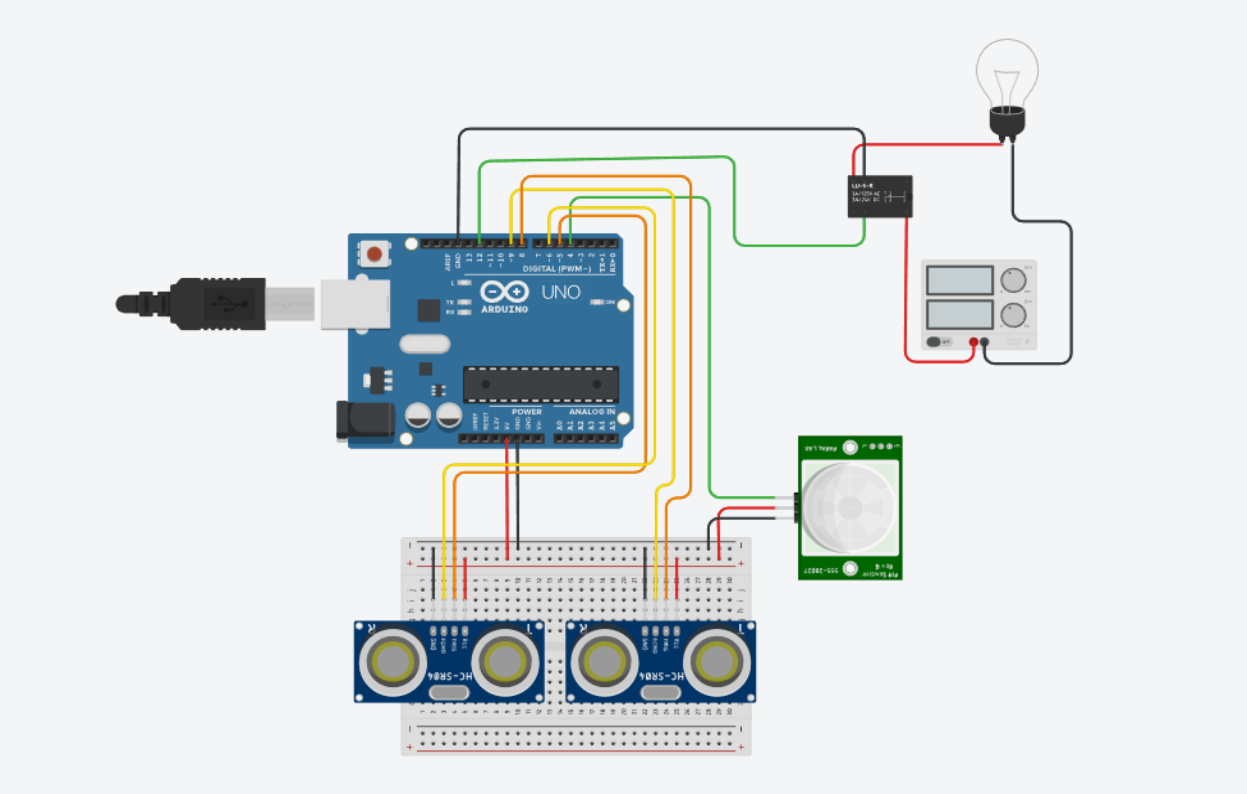
h) **Jumper Wires:**

Jumper wires are used to establish connections between the components on the breadboard. They provide a flexible and easy way to connect the various pins and components.

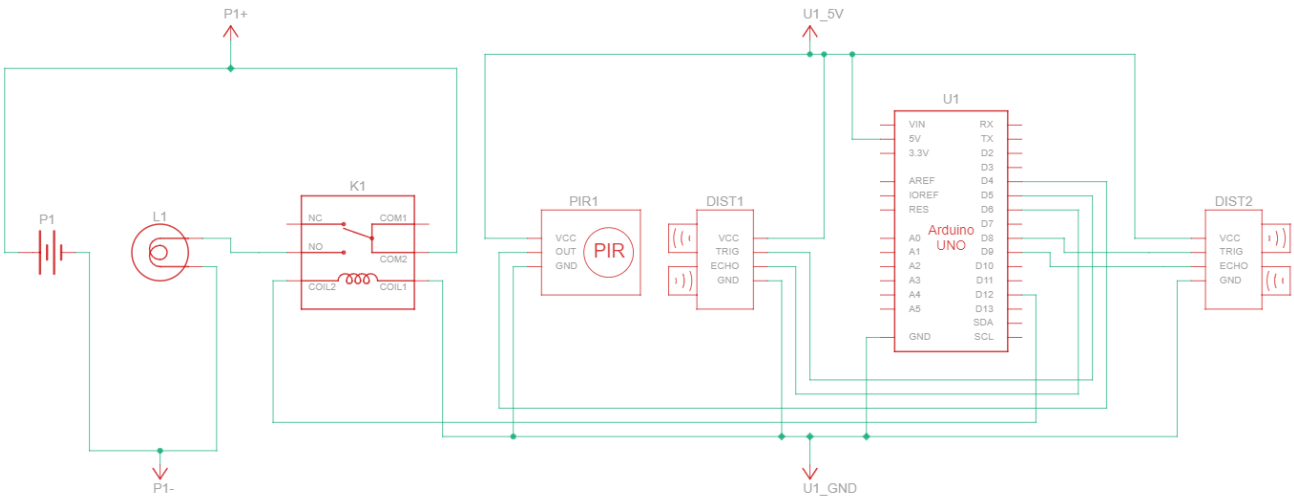
i) **Software:**

- **Arduino IDE:**
  The Arduino IDE (Integrated Development Environment) is the software used to write, compile, and upload the code to the Arduino board. It provides a user-friendly interface for programming the Arduino.
- **TinkerCAD:**
  TinkerCAD is a web-based software that allows you to simulate and design electronic circuits. It can be used to create a virtual prototype of the project before implementing it in hardware.
- **Fritzing:**
  Fritzing is a software tool used for designing and documenting electronic circuits

4. SIMULATED CIRCUIT



Schematic View

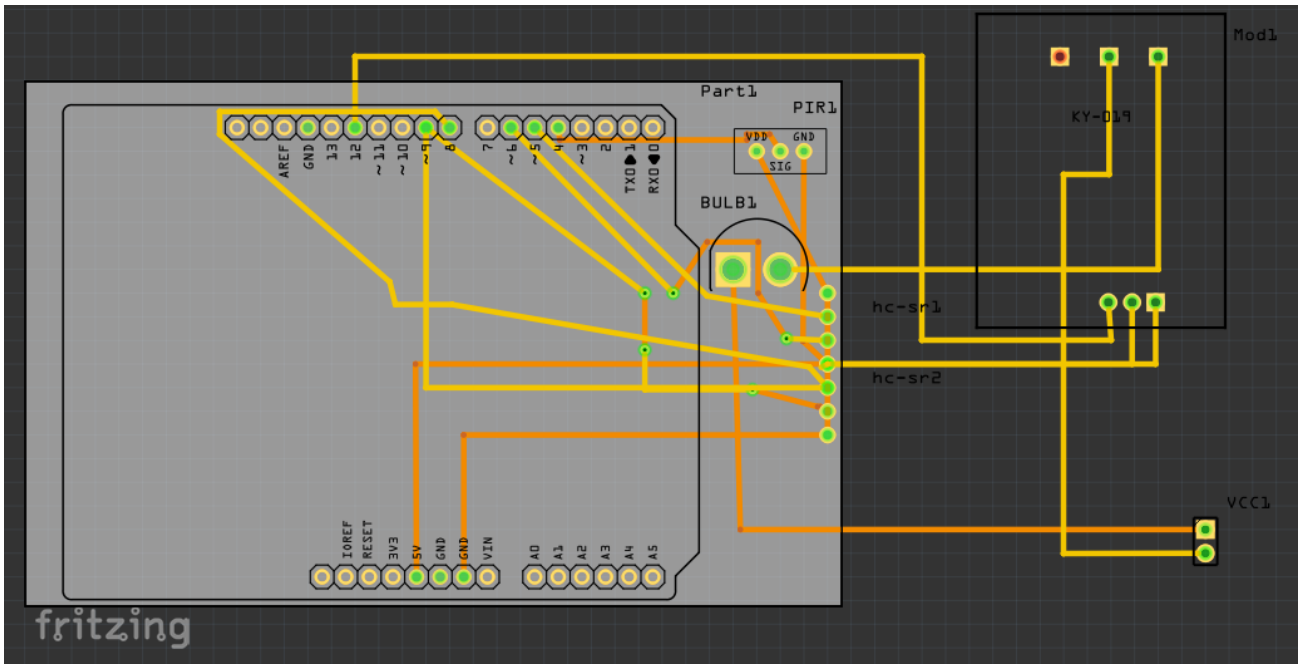


5. DEMO VIDEO

Simulation Link –

https://drive.google.com/file/d/1xWJTFBYXd4DEwbxSWPTAvi2dJDQSaa5h/view?usp=sharing

6. GERBER FILE



Gerber File Link –

https://github.com/ArunViji/MC-PROJECT/tree/7db816dcebc05f2ed18ed60f1dc3aa6d139c1120/VIT_BATCH%2028%20(1)/Gerber%20File

7. CODE

```
int PIR = 4; // PIR sensor pin
int Bulb = 12; // Bulb pin
int currentPeople = 0; // Variable to store the current number of people in
the room
#define trigPin1 5 // Ultrasonic sensor 1 trigger pin
#define echoPin1 6 // Ultrasonic sensor 1 echo pin
#define trigPin2 8 // Ultrasonic sensor 2 trigger pin
#define echoPin2 9 // Ultrasonic sensor 2 echo pin
String sequence = ""; // Variable to store the sequence of sensor readings
int timeoutCounter = 0; // Counter to track timeouts
```

```
unsigned long previousMillis = 0; // Variable to store the previous time
unsigned long previousMillis1 = 0;
const long interval = 1000; // Interval for ultrasonic sensor readings
unsigned long ultrasonicInterval = 100; // Interval for checking ultrasonic
sensors

void setup()
{
  Serial.begin(9600); // Initialize serial communication
  pinMode(PIR, INPUT); // Set PIR sensor pin as input
  pinMode(Bulb, OUTPUT); // Set Bulb pin as output
  digitalWrite(Bulb, LOW); // Turn off the Bulb initially
  pinMode(trigPin1, OUTPUT); // Set ultrasonic sensor 1 trigger pin as output
  pinMode(echoPin1, INPUT); // Set ultrasonic sensor 1 echo pin as input
  pinMode(trigPin2, OUTPUT); // Set ultrasonic sensor 2 trigger pin as output
  pinMode(echoPin2, INPUT); // Set ultrasonic sensor 2 echo pin as input
}

void loop()
{
  unsigned long currentMillis = millis(); // Get the current time

  // Read ultrasonic sensors at a fixed interval
  if (currentMillis - previousMillis >= ultrasonicInterval) {
    previousMillis = currentMillis;
    int sensor1Val = measureDistance(trigPin1, echoPin1); // Read distance
from ultrasonic sensor 1
    int sensor2Val = measureDistance(trigPin2, echoPin2); // Read distance
from ultrasonic sensor 2

    // Process the data
    if (sensor1Val < 150 && sequence.charAt(0) != '1') {
      sequence += "1"; // Add '1' to the sequence if sensor 1 detects a person
    }
    else if (sensor2Val < 150 && sequence.charAt(0) != '2') {
      sequence += "2"; // Add '2' to the sequence if sensor 2 detects a person
    }

    if (sequence.equals("12")) {
      currentPeople++; // Increment the number of people if the sequence is
'12'
      sequence = ""; // Reset the sequence
    }
    else if (sequence.equals("21") && currentPeople > 0) {
      currentPeople--; // Decrement the number of people if the sequence is
'21' and there are people in the room
      sequence = ""; // Reset the sequence
    }
```

```
    // Resets the sequence if it is invalid or timeouts
    if (sequence.length() > 2 || sequence.equals("11") ||
sequence.equals("22") || timeoutCounter > 200) {
      sequence = ""; // Reset the sequence
    }

    if (sequence.length() == 1) {
      timeoutCounter++; // Increment the timeout counter if the sequence
length is 1
    }
    else {
      timeoutCounter = 0; // Reset the timeout counter
    }

    // Print values to serial for debugging
    Serial.print("Seq: ");
    Serial.print(sequence);
    Serial.print("No. of People : ");
    Serial.println(currentPeople);
  }

  if (currentPeople > 0) {
    if (digitalRead(PIR) == HIGH) {
      digitalWrite(Bulb, HIGH); // Turn on the Bulb if there are people in the
room and motion is detected
      previousMillis1 = currentMillis; // Update the previous time
    }
    else {
      if (currentMillis - previousMillis1 <= 120000) {
        digitalWrite(Bulb, HIGH); // Keep the Bulb on for 2 minutes after
motion ceases
      }
      else {
        digitalWrite(Bulb, LOW); // Turn off the Bulb if there are no people
in the room and no motion detected
      }
    }
  }
  else {
    digitalWrite(Bulb, LOW); // Turn off the Bulb if there are no people in
the room
  }
}

// Function to measure the distance using the ultrasonic sensor
int measureDistance(int tp, int ep) {
  digitalWrite(tp, LOW);
```

```
  delayMicroseconds(2);
  digitalWrite(tp, HIGH);
  delayMicroseconds(10);
  digitalWrite(tp, LOW);
  int duration = pulseIn(ep, HIGH);
  int distance = duration * 0.034 / 2;
  return distance;
}
```

- The code begins by defining the pins for the ultrasonic sensors, PIR sensor, and the bulb. It also initializes variables for storing the sequence of sensor readings, timeout counter, and previous time.
- In the **loop()** function, the ultrasonic sensors are read at a fixed interval using the **measureDistance()** function. The distances measured by the sensors are processed to determine the presence of people in the room.
- The code checks the sequence of sensor readings and increments or decrements the count of people accordingly. It also handles invalid sequences and timeouts.
- The code then checks if there are people in the room and if motion is detected by the PIR sensor. If both conditions are met, the bulb is turned on. If there are no people in the room, the bulb is turned off. If there are people but no motion is detected for 2 minutes, the bulb is turned off as well.
- The **measureDistance()** function is used to measure the distance using the ultrasonic sensor. It sends a trigger signal, measures the duration of the echo pulse, and calculates the distance based on the speed of sound.