

## Smart SightLight

By Moritz von der Brake 31698

### Introduction by Moritz von der Brake

Many people use a night light to help them navigate their home in the dark. However there are many issues with the common night light. One issue is that a regular nightlight is static and dose not move. This means that the night light either does not have a large field of illumination or is very broad and lights up everything. The problem is made even worse by the fact that conventional nightlights attach directly to a wall socket. This is where our project the Smart Sightlight comes in. Our nightlight actively waits for a person to enter the vicinity. Once a person is in the vicinity the nightlight scans the room and calculates position and direction of movement of the person. It then uses this information to illuminate the light in-front of the person.

### Method

#### 1. CAD

In order to house all the electronics in a space friendly and secure way

### Hardware Components

| Component        | Purpose                              |
|------------------|--------------------------------------|
| ESP32            | Main microcontroller                 |
| VL53L0X          | Time-of-Flight (ToF) distance sensor |
| PIR sensors (2x) | Detect left/right motion             |
| Servos (2x)      | Rotate ToF and LED for tracking      |
| NeoPixel LED     | Light                                |

### Libraries Used

Adafruit\_VL53L0X: Controls the ToF sensor.

Adafruit\_NeoPixel: Controls the RGB LED.

ESP32Servo: PWM servo motor control on ESP32.

### Logic

At startup, the system performs an initial environmental benchmark by scanning from 0° to 180°, saving the distance data for each angle.

→Then it enters a loop waiting for PIR motion detection.

When motion is detected:

→It determines direction (left or right PIR).

→Performs a scanning sweep (with ToF) over the affected region.

If a significant difference from the benchmark is found:

→It activates a white LED.

→It moves a secondary servo to point an LED at the change.

The system re-benchmarks the environment every 5 minutes

### Code /\*\*

```
//--- include libraries ---
#include <Adafruit_VL53L0X.h>
#include <Adafruit_NeoPixel.h>
#include <ESP32Servo.h>

// --- Defines Pins ---
#define PIR_LEFT_PIN    4
#define PIR_RIGHT_PIN   5
#define LED_PIN         15
#define TOF_SERVO_PIN   17
#define LED_SERVO_PIN   16

// --- Constants ---
#define NUM_PIXELS      1
#define MAX_ANGLE        180
#define STEP_ANGLE       1
#define CHANGE_THRESHOLD 100 // mm
#define BENCHMARK_INTERVAL 300000 // 5 min in ms
#define PIR_DEBOUNCE_TIME 1000 // ms

// --- Globals ---
Adafruit_VL53L0X lox = Adafruit_VL53L0X();
Adafruit_NeoPixel pixels(NUM_PIXELS, LED_PIN, NEO_GRB + NEO_KHZ800);

Servo tofServo;
Servo ledServo;

int benchmark[MAX_ANGLE + 1];
bool scanning = false;
bool benchmarkReady = false;
int lastMotionDirection = -1; // 0 = left, 1 = right
unsigned long lastBenchmarkTime = 0;
unsigned long lastPirTime = 0;

// --- Function Prototypes ---
void lightUp(bool on);
void moveTofServo(int angle);
void moveLedServo(int angle);
void performBenchmark();
bool scanMotion(int startAngle, int endAngle);

//Initializes:
//  Serial monitor for debugging.
```

```
// PIR pins.
// ToF sensor in continuous mode.
// NeoPixel.
// Servos with appropriate PWM settings for ESP32.
// Performs an initial environmental benchmark.

void setup() {
  Serial.begin(115200);

  pinMode(PIR_LEFT_PIN, INPUT);
  pinMode(PIR_RIGHT_PIN, INPUT);

  if (!lox.begin()) {
    Serial.println(F("Failed to start VL53L0X!"));
    while (1);
  }
  lox.startRangeContinuous();

  // Servos
  ESP32PWM::allocateTimer(0);
  ESP32PWM::allocateTimer(1);
  ESP32PWM::allocateTimer(2);
  ESP32PWM::allocateTimer(3);
  tofServo.setPeriodHertz(50);
  ledServo.setPeriodHertz(50);
  tofServo.attach(TOF_SERVO_PIN, 500, 2500);
  ledServo.attach(LED_SERVO_PIN, 500, 2500);

  pixels.begin();
  pixels.setBrightness(100);
  pixels.setPixelColor(0, pixels.Color(0, 0, 0)); // Ensure off at start
  pixels.show();

  performBenchmark(); // Initial benchmark
}

void loop() {
  unsigned long now = millis();

  // Re-benchmark every 5 minutes
  if ((now - lastBenchmarkTime > BENCHMARK_INTERVAL) && !scanning) {
    performBenchmark();
  }

  // Motion detection with filtering
  // Checks if PIRs are triggered over 300ms to reduce false positives.
  // Triggers scanning only if a direction is confidently detected.
  if (!scanning && benchmarkReady) {
    if ((now - lastPirTime) > PIR_DEBOUNCE_TIME) {
```

```

bool leftTriggered = false;
bool rightTriggered = false;

// Read the PIR pins multiple times over ~300ms
for (int i = 0; i < 6; i++) {
    if (digitalRead(PIR_LEFT_PIN) == HIGH) leftTriggered = true;
    if (digitalRead(PIR_RIGHT_PIN) == HIGH) rightTriggered = true;
    delay(50); // 6 × 50ms = 300ms total sampling
}

if (leftTriggered) {
    Serial.println("PIR LEFT triggered (filtered)");
    lastMotionDirection = 0;
    scanning = true;
    lastPirTime = now;
} else if (rightTriggered) {
    Serial.println("PIR RIGHT triggered (filtered)");
    lastMotionDirection = 1;
    scanning = true;
    lastPirTime = now;
}
}

// Start scanning Sweeps back and forth. Scans until it finds two
consecutive sweeps with no changes.
if (scanning) {
    int noChangeSweeps = 0;
    int start = (lastMotionDirection == 0) ? 0 : 90;
    int end = MAX_ANGLE;

    while (noChangeSweeps < 2) {
        Serial.printf("TOF Sweep (no-change count: %d)\n", noChangeSweeps);
        bool changed = scanMotion(start, end);
        if (changed) {
            noChangeSweeps = 0;
        } else {
            noChangeSweeps++;
        }

        // Reverse sweep direction for next run
        int temp = start;
        start = end;
        end = temp;
    }

    Serial.println("No more movement. Scan finished.");
    lightUp(false);
    scanning = false;
}

```

```
}  
// Sweeps 0°–180°, storing the baseline distance values from ToF.Retries if  
distance is 0 (sensor failed)  
void performBenchmark() {  
    Serial.println("Benchmarking environment...");  
    for (int angle = 0; angle <= MAX_ANGLE; angle += STEP_ANGLE) {  
        moveToServo(angle);  
        delay(15);  
        int dist = lox.readRange();  
        if (dist == 0) {  
            angle -= STEP_ANGLE;  
            continue; // Retry this angle  
        }  
        benchmark[angle] = dist;  
        Serial.printf("Angle %d° = %d mm\n", angle, dist);  
    }  
    benchmarkReady = true;  
    lastBenchmarkTime = millis();  
    Serial.println("Benchmark complete.");  
}  
//Reads distance at each angle.  
//Compares it with the benchmark value.  
//If difference > CHANGE_THRESHOLD (100 mm): Activates LED  
//Moves LED servo to angle (inverted)  
  
bool scanMotion(int startAngle, int endAngle) {  
    bool foundChange = false;  
    int step = (startAngle < endAngle) ? STEP_ANGLE : -STEP_ANGLE;  
  
    for (int angle = startAngle; angle != endAngle + step; angle += step) {  
        moveToServo(angle);  
        delay(15);  
        int distance = lox.readRange();  
        if (distance == 0) continue;  
  
        int diff = abs(distance - benchmark[angle]);  
        if (diff > CHANGE_THRESHOLD) {  
            Serial.printf("Change at %d° → %d mm (was %d mm)\n", angle, distance,  
benchmark[angle]);  
            moveLedServo(angle);  
            lightUp(true);  
            foundChange = true;  
        }  
    }  
  
    if (!foundChange) {  
        lightUp(false);  
    }  
  
    return foundChange;  
}
```

```
void moveToServo(int angle) {
    tofServo.write(angle);
}

void moveLedServo(int angle) {
    ledServo.write(160 - angle); // Inverted to match orientation
}

void lightUp(bool on) {
    if (on) {
        pixels.setPixelColor(0, pixels.Color(255, 255, 255)); // Pure white
    } else {
        pixels.setPixelColor(0, pixels.Color(0, 0, 0));
    }
    pixels.show();
}
```

From:

<https://student-wiki.eolab.de/> - HSRW EOLab Students Wiki

Permanent link:

<https://student-wiki.eolab.de/doku.php?id=amc:ss2025:group-x:start&rev=1753532610>

Last update: **2025/07/26 14:23**

